Keywords: Monte Carlo Optimization, Black-box Optimization, Parametric Learning, Automated Annealing, Bias-variance-covariance

Parametric Learning and Monte Carlo Optimization

David H. Wolpert DHW@EMAIL.ARC.NASA.GOV
MS 269-1, Ames Research Center
Moffett Field, CA 94035.

Dev G. Rajnayaran DGORUR@STANFORD.EDU
Department of Aeronautics and Astronautics,
Durand Rm. 158, 496 Lomita Mall, Stanford, CA 94305.

Abstract

This paper uncovers and explores the close relationship between Monte Carlo Optimization of a parametrized integral (MCO), Parametric machine-Learning (PL), and 'blackbox' or 'oracle'-based optimization (BO). We make four contributions. First, we prove that MCO is mathematically identical to a broad class of PL problems. This identity potentially provides a new application domain for all broadly applicable PL techniques: MCO. Second, we introduce immediate sampling, a new version of the Probability Collectives (PC) algorithm for blackbox optimization. Immediate sampling transforms the original BO problem into an MCO problem. Accordingly, by combining these first two contributions, we can apply all PL techniques to BO. In our third contribution we validate this way of improving BO by demonstrating that cross-validation and bagging improve immediate sampling. Finally, conventional MC and MCO procedures ignore the relationship between the sample point locations and the associated values of the integrand; only the values of the integrand at those locations are considered. We demonstrate that one can exploit the sample location information using PL techniques, for example by forming a fit of the sample locations to the associated values of the integrand. This provides an additional way to apply PL techniques to improve MCO.

1. Introduction

This paper uncovers and explores some aspects of the close relationship between Monte Carlo Optimization of a parametrized integral (MCO), Parametric machine Learning (PL), and 'blackbox' or 'oracle-based' optimization (BO). We make four primary contributions. First, we establish a mathematical identity equating MCO with PL. This identity potentially provides a new application domain for all broadly-applicable PL techniques, viz., MCO.

Our second contribution is the introduction of immediate sampling. This is a new version of the Probability Collectives (PC) approach to blackbox optimization. PC encompasses Estimation of Distribution Algorithms (EDAs)(De Bonet et al., 1997; Larraaga and Lozano, 2001; Lozano et al., 2005) and the Cross Entropy (CE) method (Rubinstein and Kroese, 2004) as special cases. However PC is broader and more fully motivated. This means it uncovers (and overcomes) formal shortcomings in those other approaches.

In the immediate sampling version of PC the original BO problem is transformed into an MCO problem. In light of our first contribution, this means we can apply PL to immediate sampling. In this way *all* PL techniques — including cross-validation, bagging, boosting, active learning, stacking, and others — can be applied to blackbox optimization.

In our third contribution we experimentally explore the power of this identity between MCO and PL. In these experiments we demonstrate that cross-validation and bagging improve the performance of immediate sampling blackbox optimization. In particular, in these experiments we show that cross-validation can be used to adaptively set an 'annealing schedule' for blackbox optimization using immediate sampling *without any extra calls to the oracle*. In some cases, we show that this adaptively formed annealing schedule results in better optimization performance than *any* exponential annealing schedule.[1]

Finally, conventional MC and MCO procedures ignore the relationship between the sample point locations and the associated values of the integrand. (Only the values of the integrand at the sample locations are considered by such algorithms.) We end by exploring ways to use PL techniques to exploit the information in the sample locations, for instance, by Bayesian fitting of a surface from the sample locations to the associated values of the integrand. This constitutes yet another way of applying PL to MCO in general, and therefore to BO in particular.

1.1 Background on PL, MCO, Blackbox Optimization, and PC

We begin by sketching the four disciplines discussed in this paper:

1. A large number of parametric machine-learning problems share the following two properties. First, the goal in these problems is to find a set of parameters, θ, that minimizes an integral of a function that is parametrized by θ. Second, to help us find that θ we are are given samples of the integrand. These problems reduce to a sample-based search for the θ that we predict minimizes the integral. We will refer to problems of this class as Parametric Learning (PL) problems.

   An example of PL is parametric supervised learning, where we want to find an optimal predictor or regressor z_θ that minimizes the associated expected loss, $\int dx\, dy\;\; P(x)P(y \mid x)L[y, z_\theta(x)]$, where x's are inputs and y's are outputs. We do not, however, know $P(x)P(y \mid x)$. Instead, we are provided a training set of samples of $P(x)P(y \mid x)$. The associated PL problem is to use those samples to estimate the optimal θ.

2. MCO is a technique for solving problems of the form $\text{argmin}_\phi \int dw\, U(w, \phi)$ (see Ermoliev and Norkin, 1998). MCO starts by replacing that integral with an importance-sample generated estimate of it. That estimate is a sum parametrized by ϕ. In MCO one searches for the value ϕ that minimizes this sum; the result of this search is one's estimate of the ϕ that optimizes the original integral.

3. Blackbox optimization algorithms are ways to minimize functions of the form $G : X \to \mathbb{R}$ when one does not actually know the function G. Such algorithms work by an iterative process in which they first select a query $x \in X$, and then an 'oracle' returns to the algorithm a (potentially noise-corrupted) value $G(x)$, and no other information, in particular, no gradient information. The difference between one blackbox optimization algorithm and another is how they select each successive query based on the earlier responses of the oracle. Examples of blackbox optimization algorithms are genetic algorithms (Mitchell, 1996), simulated annealing (Kirkpatrick et al., 1983), hill-climbing algorithms, Response-Surface Methods (RSMs) (Myers and Montgomery, 2002), and some forms of Sequential Quadratic Programming (SQP) (Gill et al., 1981; Nocedal and Wright, 1999), Estimation of Distribution Algorithms (EDAs)(De Bonet et al., 1997; Larraaga and Lozano, 2001; Lozano et al., 2005), tabu search, the Cross Entropy (CE) method (Rubinstein and Kroese, 2004), and others.

1. Since they are special cases of PC, presumably we could similarly apply PL techniques to improve EDA's or the CE method.

4. PC is a set of techniques that can be used for blackbox optimization. Broadly speaking, PC works by transforming a search for the best value of a variable x into a search for the best probability distribution over the variable, $q(x)$ (see Wolpert et al., 2006; Macready and Wolpert, 2005; Wolpert, 2003, 2004a; Bieniawski and Wolpert, 2004; Antoine et al., 2004; Lee and Wolpert, 2004). Once one solves for the optimal $q(x)$, inversion to get the optimal x for the original search problem is stochastic; one simply samples q. As described below, PC has many practical strengths, and is related to RSMs, EDAs, and the CE method.

1.2 Roadmap of This Paper

We make four primary contributions:

1. Sec. 2 begins with a detailed review of MCO and PL. Conventional analysis of Monte Carlo estimation involves a bias-variance decomposition of the error of the estimator of a particular integral. We show that for MCO, a full analysis requires more than simply extending such bias-variance analysis separately to each of the estimators given by the separate ϕ's. Moments coupling the errors of the estimators for the separate ϕ's must also be taken into account. How should we do that?

   To answer this, we note that in a different context, the techniques of PL take such coupling moments into account, albeit implicitly. This leads us to explore the relation between MCO and PL. This in turn leads to our first major contribution, the proof that MCO is identical to PL. This contribution means that one can apply all PL techniques, for instance, cross-validation, bagging, boosting, stacking, active learning and others, to MCO. Such PL-based MCO (PLMCO) provides a new way of minimizing potentially high-dimensional parametrized integrals.

   Experimentally testing the utility of applying PL to MCO requires an MCO application domain. Here we choose the domain of blackbox optimization. To establish how blackbox optimization is an application domain for MCO requires our second contribution, as follows.

2. We start in Sec. 3 by presenting an overview of previous versions of the blackbox optimization approach of PC. We then make our second contribution in the following section, where we introduce immediate sampling, a new version of PC that overcomes some of the limitations of previous versions.

   These first two contributions are combined by the fact that immediate sampling is a special case of MCO. The resultant identity between PL and immediate sampling means that, in principle, any PL technique can be applied to blackbox optimization. In particular, regularization, cross-validation, bagging, active learning, boosting, stacking, kernel machines, and others, can be 'cut and paste' to do blackbox optimization. This use of PL for blackbox optimization constitutes a new application domain for PL.

   In Sec. 4.5 we present some concrete instances of how to modify immediate sampling to use PLMCO rather than conventional MCO. It is important to note that when applied (via immediate sampling) to blackbox optimization, these PL techniques do *not* require additional calls to the oracle. For example, using cross-validation to set regularization parameters in immediate sampling (the equivalent of an annealing schedule in SA) does not involve running the entire blackbox optimization algorithm with different regularization schedules. As another example, using bagging in immediate sampling does not mean running the optimization algorithm multiple times based on different subsets of the sample points found so far.

3. Our third contribution is to experimentally demonstrate in Sec. 5 that PLMCO substantially outperforms conventional MCO when used this way for blackbox optimization. We are particularly interested in blackbox optimization problems where calls to the oracle are the primary

expense. Accordingly, non-oracle, 'offline' computation is considered free. So in our experiments we compare algorithms based on the values of G found by the algorithms versus the associated number of calls to the oracle[2]. In particular, we show that bagging and cross-validation leads to faster blackbox optimization on two well-known benchmark problems for continuous nonconvex optimization.

It should be emphasized that these experiments are *not* intended to investigate whether PLMCO applied to immediate sampling is superior to other blackbox optimization algorithms. Rather their purpose is to investigate whether one can indeed leverage the formal connection between PL and MCO to improve immediate sampling. Accordingly, these experiments are on toy domains, and we do not compare performance with other blackbox optimization algorithms. We leave such comparisons to future papers.

4. In estimating the value of an integral based on random samples of its integrand, conventional MC and MCO techniques ignore how the locations of the sample points are related to the associated values of the integrand. Such techniques concentrate exclusively on those sample values of the integrand that are returned by the oracle. However, one can use the sample locations and associated integrand values to form a supervised learning fit to the integrand. In principle, such a fit can then be used to improve the overall estimate of the integral.

   In 'fit-based' MC and MCO one uses all the data at hand to fit the integrand and then uses that fit to improve the algorithm. In this paper, we concentrate on situations where the data at hand consist only of sample locations and the associated values of the integrand, but in other situations the data at hand may also include information like the gradient of the integrand at the sample points. In their most general form, fit-based MC and MCO include techniques to exploit such information.

   One natural Bayesian approach to fit-based MC uses Gaussian processes. Work adopting this approach, for the case where the data only contain sample locations and associated integrand values, is reviewed in Rasmussen and Gharamani (2003). In Sec. 6 we generalize that work on fit-based MC, e.g., to allow non-Bayesian approaches. In that section, we also consider fit-based MCO in general, and fit-based immediate sampling in particular.

One of the ways cross-validation is used in these experiments is to set a regularization parameter. In immediate sampling, the regularization parameter plays the same role as the temperature does in simulated annealing. So intuitively speaking, our results show how to use cross-validation to set an annealing schedule adaptively for blackbox optimization, *without extra calls to the oracle*. We show in particular that such auto-annealing outperforms the best-fit exponential annealing schedule.

There are more topics involving the connection between MCO, immediate sampling and PL, than can be explored in this single paper. One such topic is how to incorporate constraints on x in immediate sampling. Another important topic involves a derivation from first principles of the objective function used in immediate sampling. These two topics are briefly discussed in the appendices. Some other topics are mentioned, albeit even more briefly, in the conclusion.

1.3 Notation

As a point of notation, we will use the term 'distribution' to refer either to a probability distribution or a density function, with the associated Borel field implicitly fixing the meaning. Similarly, we will write integrals even when we mean sums; the measure of the

2. See Wolpert and Macready (1997); Droste et al. (2002); Wolpert and Macready (2005); Corne and Knowles (2003); Igel and Toussaint (2004); Schumacher et al. (2001) for a discussion of the mathematics relating algorithms under such performance measures.

integral is implicitly taken to be the one appropriate for the the argument. We will use Θ to indicate the Heaviside or indicator function, which is 1 if its argument is positive, and 0 otherwise.

We will use $\mathscr{P}$ to mean the set of all distributions over X. We are primarily interested in X's that are too large to permit computations involving all members of $\mathscr{P}$. Accordingly, we will will work with parametrized subsets $\mathscr{Q} \subset \mathscr{P}$. We generically write that (possibly vector-valued)parameter as θ, and write the element of $\mathscr{Q}$ specified by θ as q_θ. We use $\mathbb{E}$ to indicate the expectation of a random variable. Subscripts on $\mathbb{E}$ are sometimes used to indicate the distribution(s) defining the expectation.

We take any oracle $\mathscr{G}$ to be an x-indexed set of independent stochastic processes, and use the symbol g to indicate the generic output of the oracle in response to any query. With some abuse of notation, we denote the output of the oracle for query x as $P(g \mid x, \mathscr{G})$. For a noise-free, or single-valued oracle, we write $P(g \mid x, \mathscr{G}) = \delta(g - G(x))$ for some function G implicitly specified by $\mathscr{G}$, where $\delta(.)$ is the Dirac delta function.

When there is both a factual version of a random variable and a posterior distribution over counter-factual values of that variable, they must be distinguished. In general this requires extending the conventional Bayesian formalism (see Wolpert, 1997, 1996). Here, though, it suffices for us to indicate counter-factual values by a subscript c. Say there is a factual oracle $\mathscr{G}$, and we are provided a data set D formed by sampling $\mathscr{G}$. We use superscripts to denote different samples in that data set. Then D in turn induces a posterior over oracles, and we write that posterior as $P(\mathscr{G}_c \mid D)$.

2. MCO and PL

In this section we review PL and MCO show that they are mathematically identical.

2.1 Overview of PL

A broad class of parametric machine learning problems try to find

$$\text{(P1)}: \qquad \text{argmin}_\xi \int dx \; P(x) R_x(\xi).$$

For subsequent purposes, it will be useful to write x as a subscript and ξ as an argument of R, even though x is the integration variable and ξ is the parameter being optimized. To perform this minimization, we have a set of function values $D \equiv \{R_{x^i}(\xi)\}$, where we typically assume that the samples $x^i, i = 1, \ldots, N$ were formed by IID sampling of $P(x)$.

The maximum likelihood approach to this minimization first makes the approximation

$$\begin{aligned} \int dx \; P(x) R_x(\xi) &\approx \frac{1}{N} \sum_i R_{x^i}(\xi), \\ &\triangleq \frac{1}{N} \sum_i R^i(\xi). \end{aligned} \tag{1}$$

One then solves for the ξ minimizing the sum, and uses this as an approximation to the solution to P1. In practice, though, this procedure is seldom used directly: although the approximation in Eq. 1 is unbiased for any fixed ξ, $\min_\xi \sum_i R^i(\xi)$ is not an unbiased estimate of $\min_\xi \int dx \; P(x) R_x(\xi)$. Therefore, when this approximation is exploited, it is modified to incorporate bias-reduction techniques.

EXAMPLE: PARAMETRIC SUPERVISED LEARNING:

Let X, Y be input and output spaces, respectively. Let $L(y^1, y^2) : Y \times Y \to \mathbb{R}$ be a loss function, and $z_\xi : X \to Y$ be a ξ-parametrized set of functions. In parametric machine learning with IID error our goal is to solve

$$\begin{aligned} &\text{argmin}_\xi \quad \int dx\ P(x) \int dy\ P(y \mid x) L(y, z_\xi(x)), \\ &\qquad \triangleq \\ &\text{argmin}_\xi \quad \int dx\ P(x) R_x(\xi). \end{aligned} \tag{2}$$

Intuitively, $R_x(\xi)$ is the expected loss at x for the 'fit' $z_\xi(x)$ to the x-indexed set of distributions $P(y \mid x)$.

To perform this minimization we have a **training set** of pairs $D \equiv \{x^i, y^i\}, i = 1, \ldots, N$, that we assume were formed by IID sampling of $P(x)P(y \mid x)$. The maximum likelihood approach to this minimization first makes the approximation

$$\begin{aligned} \int dx\ P(x) \int dy\ P(y \mid x) L(y, z_\xi(x)) &\approx \frac{1}{N} \sum_i L(y^i, z_\xi(x^i)), \\ &\triangleq \frac{1}{N} \sum_i R^i(\xi). \end{aligned}$$

One then solves for the ξ minimizing the sum, and uses this as an approximation to the solution to (P1). As discussed above, in practice, this minimization is rarely used directly, and is usually combined with a bias-reducing technique like cross-validation.

2.2 Overview of MCO

Consider the problem

$$\text{(P2)}: \qquad \text{argmin}_{\phi \in \Phi} \int dw\ U(w, \phi).$$

For now, we do not impose constraints on ϕ, nor restrict Φ. Monte Carlo Optimization (Ermoliev and Norkin, 1998) is a way to search for the solution of (P2). In MCO we use importance sampling to rewrite the integral in (P2) as

$$\begin{aligned} \int dw\ U(w, \phi) &= \int dw\ v(w) \frac{U(w, \phi)}{v(w)}. \\ &\triangleq \int dw\ v(w) r_{v,U,w}(\phi), \end{aligned} \tag{3}$$

for some sampling distribution v. Following the usual importance sampling procedure, we IID sample v to form a sample set $\{U(w^i, .) : i = 1, \ldots N\}$, which specifies a set of N sample functions

$$r^i(\phi) \triangleq r_{v,U,w^i}(\phi).$$

It is implicitly assumed that for any w, we can evaluate $v(w)$ up to an overall normalization constant.

In MCO, these N functions are used in combination with any prior information to estimate the solution to (P2). Conventionally, this is done by approximating the solution to (P2) with the solution to the problem

$$\text{(P3)}: \qquad \text{argmin}_\phi \sum_i r^i(\phi).$$

We define

$$\begin{aligned} \mathscr{L}_U(\phi) &\triangleq \int dw\, U(w,\phi), \\ \hat{\mathscr{L}}_{v,U,\{w^i\}}(\phi) &\triangleq \sum_i r_{v,U,w^i}(\phi), \\ \hat{\phi}_{v,U,\{w^i\}}(\phi) &\triangleq \text{argmin}[\hat{\mathscr{L}}_{v,U,\{w^i\}}(\phi)]. \end{aligned}$$

For notational simplicity, the subscripts will usually be omitted in these expressions. We will use the term **naive MCO** to refer to solving (P3) by minimizing $\hat{\mathscr{L}}(\phi)$.

2.3 Statistical Analysis of MCO

The statistical analysis of MC estimation of integrals is a relatively mature field (see Robert and Casella, 2004; Fishman, 1996). We now show that when such MC estimation is combined with parameter optimization in MCO, the analysis becomes much more involved.

2.3.1 Review: MC Estimation

First consider MC estimation, with no mention of MCO. We first need to to specify a loss function $L(.,.)$ that will couple our mathematics with real-world costs. The first argument of such an L is the output of the estimation algorithm under consideration. The second argument is the quantity statistically sampled by that algorithm. The associated value of L is the cost if the algorithm produces the output specified in that first argument, using the quantity specified in the second argument.

As an example, consider importance-sampled MC estimation of an integral. Using the MCO notation just introduced, we use $\hat{\mathscr{L}}(\phi)$ as an estimate of $\mathscr{L}(\phi)$ for some fixed ϕ. The quantity being sampled is the function $U(.,\phi)$, and the output of the algorithm is $\hat{\mathscr{L}}(\phi)$. Accordingly, these are the arguments of the loss function.

The most popular loss function in statistical analysis of MC integral estimation is quadratic loss, given below.

$$L(\hat{\mathscr{L}}(\phi), U(.,\phi)) \triangleq [\hat{\mathscr{L}}(\phi) - \int dw\, U(w,\phi)]^2.$$

Unless explicitly stated otherwise, we will henceforth use the term 'expected loss' to refer to the average of this loss function over sample sets. Since $\hat{\mathscr{L}}(\phi)$ is an unbiased estimate of $\mathscr{L}(\phi)$, the expected loss is the sample variance,

$$\begin{aligned} \text{Var}(\hat{\mathscr{L}}(\phi)) &= \mathbb{E}([\sum_{j=1}^N \frac{U(w^j,\phi)}{Nv(w^j)}]^2) - [\mathbb{E}(\sum_{j=1}^N \frac{U(w^j,\phi)}{Nv(w^j)})]^2 \\ &= \frac{1}{N}\{\int dw\, v(w)[\frac{U(w,\phi)}{v(w)}]^2 - [\int dw\, v(w)\frac{U(w,\phi)}{v(w)}]^2\} \\ &= \frac{1}{N}\{\int dw\, v(w)[\frac{U(w,\phi)}{v(w)}]^2 - [\mathscr{L}(\phi)]^2\}. \end{aligned}$$

This expansion for the sample variance is quite useful. For example, one can solve for the v that minimizes this variance (and therefore minimizes expected loss) as a function of $U(., \phi)$. For nowhere-negative U, that optimal v is given by (see Robert and Casella, 2004)

$$v(w) \triangleq \frac{U(w, \phi)}{\int dw' U(w', \phi)}.$$

Given the formula for the optimal v, one can estimate it from a current sample set, and then use the estimated optimal v for future sampling. This is what is done in the VEGAS Algorithm (Lepage, 1978, 1980). Consideration of the sample variance has also led to algorithms that partition X and then run importance sampling on each partition element separately, for instance, stratified sampling (Fishman, 1996). MC estimators that do not use strict importance sampling may introduce bias. However, if the variance is sufficiently reduced, expected quadratic error is reduced. This can be exploited to tradeoff bias and variance.

2.3.2 From MC to MCO

When we combine MC with parameter optimization in MCO, quantities like $\mathrm{Var}(\hat{\mathscr{L}}(\phi))$ for one particular ϕ are not the main objects of interest. Instead, we are interested in expected loss of our iterated MCO algorithm, which involves multiple ϕ's. So what is the appropriate loss function for analyzing MCO? From the very definition of (P2), it is clear that we want $L(\phi, U)$ to be minimized by the ϕ that minimizes $\int dw\, U(w, \phi)$. The simplest approach to doing this, which will be assumed from now on, stipulates that

$$L(\phi, U) \;=\; \mathscr{L}(\phi) \;=\; \int dw\, U(w, \phi), \tag{4}$$

the same integral appearing in (P2). If we can solve (P2) exactly, then we will have produced the ϕ with minimal value of this loss function.

Given this choice of loss function, expected loss in naive MCO is

$$\begin{aligned}
\mathbb{E}(L \mid U, v) &= \int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i) \mathscr{L}(\mathrm{argmin}_\phi[\hat{\mathscr{L}}_{v,U,\{w^i\}}(\phi)]) \\
&= \int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i) \int dw'\, U(w', \mathrm{argmin}_\phi[\sum_{j=1}^{N} \frac{U(w^i, \phi)}{v(w^i)}]).
\end{aligned} \tag{5}$$

The optimal v for naive MCO is the one that minimizes $\mathbb{E}(L \mid U, v)$. There is no direct relation between this v and the one that minimizes loss for some single ϕ. In stark contrast to the MC analysis in Sec. 2.3.1, in addition to the sample variance $\mathrm{Var}(\hat{\mathscr{L}}(\phi))$ for any *single* ϕ, the expected loss $\mathbb{E}(L \mid U, v)$ now also depends on moments coupling the distributions of $\hat{\mathscr{L}}(\phi)$ for *different* ϕ's. Loosely speaking, the bias-variance tradeoff in Sec. 2.3.1 now becomes a more complicated bias-variance-covariance tradeoff. Now, setting w is more involved, but we can approach it as follows.

Expressing the expected loss slightly differently gives us an important insight. Note that each sample set $\{w^i\}$ gives rise to an associated set of estimates for all $\phi \in \Phi$. Call this (possibly infinite dimensional) vector of estimates $\vec{l}$, each of whose components is indexed by ϕ and is an estimate for that particular ϕ. Now, instead of computing expected loss by averaging over all possible sample sets, we average over all possible vectors $\vec{l}$. In order to do this, we need to specify the probability of each vector $\vec{l}$. Define

$$\pi_{v,U,\Phi}(\vec{l}) \;\triangleq\; \Pr(\{w^i\} : \hat{\mathscr{L}}_{v,U,\{w^i\}}(\phi) = l_\phi \;\; \forall \phi \in \Phi).$$

So, $\pi_{v,U,\Phi}(\vec{l})$ is the probability of a set of sample points $\{w^i\}$ such that for each $\phi \in \Phi$, the associated empirical estimate $\sum_i r_{v,U,w^i}(\phi)$ equals the corresponding component of $\vec{l}$. For notational simplicity the subscripts of $\pi_{v,U,\Phi}$ will sometimes be omitted. We can now write Eq. 5 succinctly as

$$\mathbb{E}(L \mid U, v) \quad = \quad \int d\vec{l}\, \pi(\vec{l})\, \mathscr{L}(\text{argmin}_\phi[l_\phi]) \tag{6}$$

where 'argmin$_\phi[l_\phi]$' means the index ϕ of the smallest component of $\vec{l}$. The **risk** is the difference between this expected loss and the lowest possible loss. We can write that risk as

$$\int d\vec{l}\, \pi_{v,U,\Phi}(\vec{l})\, [\mathscr{L}(\text{argmin}_\phi[l_\phi]) \; - \; \text{min}_\phi[\mathscr{L}(\phi)]]. \tag{7}$$

Our sample set constitutes a set of samples of $\pi_{v,U,\Phi}$ occurring in Eq. 6,This fact can potentially be exploited to dynamically modify v and/or Φ to reduce $\mathbb{E}(L \mid U, v)$. Indeed, for the simpler case of MC estimation, this is essentially the kind of computation done in the VEGAS algorithm mentioned above. As a practical issue, it may be difficult to update v and/or Φ using the full formula Eq. 5. Instead, one could approximate that formula $\mathbb{E}(L \mid U, v)$ near a single ϕ of interest, e.g., about a current estimate for the optimal ϕ.

Intuitively though, one would expect that for a fixed set of ϕ's, everything else being equal, it would be advantageous to have small variances of unbiased estimators and large covariances between them. Such considerations based on the second moments may help one choose quantities like the sampling distribution v.

Such considerations may also help one choose the set of candidate ϕ's, Φ. For example, one way to have large covariances between the $\phi \in \Phi$ is to have the associated functions over w, $\{U(., \phi) : \phi \in \Phi\}$, all lie close to one another in an appropriate function space (e.g., according to an l_∞ norm comparing such functions). However, choosing such a Φ will tend to mean there is a small 'coverage' of that set of functions, $\{U(., \phi) : \phi \in \Phi\}$. More precisely, it will tend to prevent the best of those ϕ's from being very good; $\text{min}_{\phi \in \Phi}[\int dw\, U(w, \phi)]$ will not be very low.

This illustrates that, in choosing Φ, there will be a tradeoff between two quantities: The first quantity is the best possible performance with any of the $\phi \in \Phi$. The second quantity is the risk, that is, how close a given MCO algorithm operating on Φ is likely to come to that best possible performance of a member of Φ. Choosing Φ to have large covariances of the (MC estimators based on the) members of Φ, and in particular to have large covariances with the truly optimal ϕ, $\text{argmin}_{\phi \in \Phi} \mathscr{L}(\phi)$, will tend to result in low risk. But it will also tend to result in poor best-possible performance over all $\phi \in \Phi$.

Similarly, one would expect that as the size of Φ increases, there would be a greater chance that a sample set for one of the suboptimal $\phi \in \Phi$ would have low expected loss 'by luck'. This would then mislead one into choosing that suboptimal ϕ. So increasing the size of Φ may increase risk. However increasing Φ's size should also improve best possible performance. So again, we get a tradeoff.

It may be that such considerations involving the size of Φ and the covariances of its members can be encapsulated in a single number, giving an 'effective size' of Φ (somewhat analogously to the VC dimension of a set of functions). Such tradeoffs are specific to the use of MCO, and do not arise in plain (single-ϕ) MC. They are in addition to the usual bias-variance tradeoffs, which still apply to each of the separate MC estimators.

An illustrative example of the foregoing is provided in App. C. A more complete statistical analysis of risk in MCO, including Bayesian considerations, is in Sec. 6.

MCO	PL
w	x
ϕ	ξ
$v(w)$	$P(x)$
$r_{v,w}(\phi)$	$R_x(\xi)$
$r_v^i(\phi)$	$R^i(\xi)$

Table 1: Correspondence between PL and MCO.

2.4 PL Equals MCO

In MCO, we have to extrapolate from the sample set of w values to perform the integral minimization in Eq. 3. As discussed above, this can recast as having a set of sample functions $\phi \rightarrow r^i(\phi)$ that we want to use to estimate the ϕ that achieves that minimization. Similarly, in PL, we have to extrapolate from a training set of functions $R^i(\xi)$ to minimize the integral $\int dx \; P(x) R_x(\xi)$. Though not usually viewed this way, at the root of this extrapolation problem is the problem of using the sample functions $\xi \rightarrow R^i(\xi)$ to estimate the minimizer of Eq. 2.

In addition, the analysis of Sec. 2.3 is closely related to the PL field of uniform convergence theory. That field can be cast in the terms of the current discussion as considering a broad class of U's, $\mathscr{U}$. Its starting point is the establishment of bounds on how

$$\max_{v, U \in \mathscr{U}} [\int d\vec{l} \, \pi_{v,U,\Phi}(\vec{l}) \, \Theta(\mathscr{L}(\text{argmin}_\phi[l_\phi]) \; - \; \min_\phi[\mathscr{L}(\phi)] \; - \; \kappa) \tag{8}$$

depends[3] on κ. Of particular interest is how the function taking κ to the associated bound varies with characteristics of $\mathscr{U}$ and Φ (see Vapnik, 1982, 1995). Eq. 8 should be compared with Eq. 7.

All of this suggests that the general MCO problem of extrapolation from a sample set of empirical functions to minimize the integral of Eq. 3, is, in fact, identical to the general PL problem of extrapolation from a training set of empirical functions to minimize the integral of Eq. 2. This is indeed the case. As shown in Table 1, identify $\xi \leftrightarrow \phi, x \leftrightarrow w, P(x) \leftrightarrow v(w), R_x(\xi) \leftrightarrow r_{v,w}(\phi), r_v^i(\phi) \leftrightarrow R^i(\xi)$. Then the integrals in Eq. 3 and (P1) become identical. So the MCO expected loss function in Eq. 4 becomes identical to the PL expected loss. Similarly, the sample functions for MCO and PL become identical.

In particular, in supervised learning, when there is no noise, $P(y \mid x)$ becomes a single-valued function $y(x)$, and the parametric supervised learning problem becomes

$$\text{argmin}_\xi \int dx \; P(x)[L(y(x), z_\xi(x))]$$

This should be compared to the MCO problem as formulated in Eq. 3. For the same reasons that direct minimization of Eq. 1 is seldom used in PL, we now see that naive MCO will be biased, and should preferably not be used directly.

Note that most sampling theory analysis of PL does not directly consider the biases and variances of the separate Monte Carlo estimators for each ξ, nor does it directly consider the moments that couple the distributions of those estimates. Rather, it considers a different

3. As an example, rewrite $w \rightarrow x, \phi \rightarrow \alpha, v(x) \rightarrow P(x)$. Also choose $\mathscr{U}$ to be all functions of the form $U(w, \phi) = U(x, \alpha) \triangleq \int dy \; P(y \mid x)(y - F(x, \alpha))^2$ for any function F and distribution $P(y \mid x)$. Under this substitution, Eq. 8 becomes the archetypal uniform convergence theory problem for regression with quadratic loss.

type of bias and variance — the bias and variance of an entire algorithm that chooses a ξ based on associated MC estimates of expected loss (Wolpert, 1997). In this sense, such PL analysis bypasses the issues considered in Sec. 2.3. The bias-variance-covariance approach described in this section might have important implications on PL analysis of learning algorithms, but for the moment, in our exploration of the identity between MCO and PL, we simply use PL-based techniques to reduce the bias or variance of our algorithms.

3. Review of PC

This section cursorily reviews the previously investigated type of PC. It then briefly discusses the advantages of PC for blackbox optimization and its relation to other blackbox optimization techniques.

3.1 Introduction to PC

To introduce PC, consider the general (not necessarily blackbox) optimization problem

$$\text{(P4)}: \qquad \text{argmin}_{x \in X} \mathbb{E}(g \mid x, \mathscr{G}).$$

For now, we ignore constraints on x. In PC we transform (P4) into the problem

$$\text{(P5)}: \qquad \text{argmin}_{q_\theta \in \mathscr{Q}} \mathscr{F}_{\mathscr{G}}(q_\theta),$$

for some appropriate function $\mathscr{F}_{\mathscr{G}}$. After solving (P5) we stochastically invert q_θ to get an x (the ultimate object of interest), by sampling q_θ. This type of "randomizing transform" contrasts with conventional transform techniques, where inversion is deterministic.

Ideally, $\mathscr{F}_{\mathscr{G}}$ should be chosen in a first-principles manner, based on exactly how q_θ will be sampled and how those samples used (see Sec. 6). In practice though, computational considerations might lead one to choose $\mathscr{F}_{\mathscr{G}}$ heuristically. Intuitively, such considerations might compel us to choose $\mathscr{F}_{\mathscr{G}}$ both so that (P5) is readily easy to solve, and so that any solution q_θ to (P5) is concentrated about the solutions of (P4). Taking the parametrization to be implicit, we often abbreviate $\mathscr{F}_{\mathscr{G}}(q_\theta)$ as just $\mathscr{F}_{\mathscr{G}}(\theta)$.

In many variants of PC explored to date, $\mathscr{F}_{\mathscr{G}}(\theta)$ is an integral transform[4] over X,

$$\mathscr{F}_{\mathscr{G}}(\theta) \triangleq \int dx\, dg\; P(g \mid x, \mathscr{G}) F(g, q_\theta(x)). \tag{9}$$

$$\triangleq \int dx\; r_{P(g|x,\mathscr{G})}(x, \theta) \tag{10}$$

As an example of such an integral transform, consider optimization with a noise-free (single-valued) oracle, $P(g \mid x, \mathscr{G}) = \delta(g - G(x))$, where the transformed objective is the expected value of $(g|x)$ under $x \sim q_\theta$. In other words, $\mathscr{F}_{\mathscr{G}} = \mathbb{E}_{q_\theta}[G(x)]$. In addition, suppose that $\mathscr{Q} = \mathscr{P}$, that is, q_θ can be *any* distribution. Under fairly weak assumptions, it can be shown that one solution to (P5) is given by the point-wise limit of Boltzmann distributions,

$$p^\star(x) = \lim_{\beta \to \infty} p^\beta(x), \text{ where } p^\beta(x) \propto \exp[-\beta G(x)].$$

In the case where $\mathscr{Q} \subset \mathscr{P}$, we could choose $\mathscr{F}_{\mathscr{G}}(\theta)$ to be a measure of the dissimilarity between such a Boltzmann (or other) 'target' distribution, and a given q_θ. For instance,

4. An instance where this is not the case is with the elite objective function, described in App. B.

we could use a Kullback-Leibler (KL) divergence between q_θ and p^β, which we refer to as "pq" KL distance:

$$\begin{aligned} \mathscr{F}_{\mathscr{G}}(\theta) &= \mathrm{KL}(p^\beta \mid\mid q_\theta) \\ &\triangleq -\int dx\; p^\beta(x) \ln[\frac{q_\theta(x)}{p^\beta(x)}]. \end{aligned}$$

In terms of the quantities in Eq. 9, $F(g, q_\theta(x)) \propto e^{-\beta g} \ln[q_\theta(x)]$, up to an overall additive constant. So $r_{P(g|x,\mathscr{G})}(x, \theta)$ in Eq. 10 is the contribution to the KL distance between p^β and q_θ given by the argument x.

To see why this choice of $\mathscr{F}_{\mathscr{G}}(\theta)$ is reasonable, first note that $p^\beta(x)$ is large where $G(x)$ is small. Indeed, as $\beta \to \infty$, p^β becomes a delta function about the x(s) minimizing $G(x)$, that is, about the solution(s) to (P4). Now, suppose that $\mathscr{Q}$ is a broad enough class that it can approximate any sufficiently peaked distribution. That means that there is a $q_\theta \in \mathscr{Q}$ for which $\mathrm{KL}(p^\beta \mid\mid q_\theta)$ is small for large β. In such a situation, the q_θ solving (P5) will be highly peaked about the x(s) solving (P4). Accordingly, if we can solve (P5) for large β, sampling the resultant q_θ will result in an x with a low $\mathbb{E}(g \mid x, \mathscr{G})$.

3.2 Review of Delayed Sampling

We now present a review of conventional, **delayed-sampling** PC. In this type of PC we exploit characteristics of the parametrization of q_θ, and pursue the *algebraic* solution of (P5) as far as possible, in closed form. At some point, if there remain quantities in this algebraic expression that we cannot evaluate closed-form, we estimate them using Monte Carlo sampling.

As an example, consider a noise-free oracle, and instead of pq Kullback-Leibler distance, choose $\mathscr{F}_{\mathscr{G}}(q_\theta)$ to be the expected value returned by the oracle under q_θ,

$$\begin{aligned} \mathbb{E}_{q_\theta,\mathscr{G}}(g) &\triangleq \int dx\, dg\; g P(g \mid x, \mathscr{G}) q_\theta(x) \\ &= \int dx\; G(x) q_\theta(x) \end{aligned} \tag{11}$$

where the second equality reflects the fact that we are assuming a noise-free oracle. To emphasize the fact that we're considering noise-free oracles[5], we will sometimes write $\mathbb{E}_{q_\theta,\mathscr{G}}(g) = \mathbb{E}_{q_\theta,G}(g)$. While $\mathbb{E}_{q_\theta,\mathscr{G}}(g)$ is a linear function of q_θ, in general it will not be a linear function of θ. Accordingly, finding the θ minimizing $\mathbb{E}_{q_\theta,\mathscr{G}}(g)$ may be a non-trivial optimization problem.

Since q_θ must be a probability distribution, (P5) is actually a constrained optimization problem, involving $|X|$ inequality constraints $\{q_\theta(x) \geq 0\ \forall x\}$, and one equality constraint, $\int dx\; q_\theta(x) = 1$. As discussed by Wolpert et al. (2006), such a constrained optimization problem can be converted into one with no inequality constraints by the use of barrier function methods. These methods transform the original optimization problem into a sequence of new optimization problems, $\{(\mathrm{P5})^i\}$, each of which is easier to solve than the original problem (P5). Solving those problems in sequence leads to a solution to the original problem (P5).

Consider applying this method with an entropic barrier for the case where $\mathscr{F}_{\mathscr{G}}(q_\theta) = \mathbb{E}_{q_\theta,G}(g)$. Then, it turns out that up to additive constants, each problem $(\mathrm{P5})^i$ is again

5. Even though it is noise-free, the oracle G may be a random variable — we may not know G, and may attempt to predict it probabilistically from data, in a Bayesian fashion. In such a situation, notation like '$\mathbb{E}(\mathscr{G})$' refers to the expected oracle under our prior distribution over oracles. So, we use $\mathbb{E}_{q_\theta,G}(g)$ rather than $\mathbb{E}_{q_\theta}(G)$, even though the latter is the notation we used in previous work on PC.

of the form of (P5). However the $\mathscr{F}_{\mathscr{G}}(q_\theta)$ of each problem $(\text{P5})^i$ is the 'qp' KL distance, $\text{KL}(q_\theta \mid\mid p^{\beta_i})$, where β_i is the value of the 'barrier parameter' specifying problem $(\text{P5})^i$. In other words, up to irrelevant additive constants, each $(\text{P5})^i$ is the problem of finding the θ that minimizes

$$\mathscr{F}_{G,\beta_i}(q_\theta) \quad = \quad \mathbb{E}_{q_\theta,G}(g) - {\beta_i}^{-1} S(q_\theta)$$

where $S(.)$ is conventional Shannon entropy[6]. In this case the barrier function method directs us to iterate the following process: Solve for the q_θ that minimizes $\text{KL}(q_\theta \mid\mid p^{\beta_i})$, and then update β_i. At the end of this process we will have a local solution to (P5).

In the case where X is a Cartesian product, we often use distributions parametrized as a product distribution, $q_\theta = \prod_i q_i(x_i)$. Under this parametrization each problem $(\text{P5})^i$ can be solved by gradient descent, where the gradient components of $\mathscr{F}_{G,\beta_i}(q_\theta)$ are given by

$$\frac{\partial \mathscr{F}_{G,\beta_i}(q_\theta)}{\partial q_i(x_i)} \quad = \quad \mathbb{E}_{q_\theta,G}(g \mid x_i) + {\beta_i}^{-1} \ln[q_i(x_i)] + \lambda_i$$

where the Lagrange parameters λ_i enforce normalization of each q_i.

There are many better alternatives[7] to simple gradient descent for minimizing each $\mathscr{F}_{G,\beta_i}(q_\theta)$, involving Newton's method, block relaxation, and related techniques (Wolpert et al., 2006). In all such schemes investigated to date, we need to repeatedly evaluate terms like $\mathbb{E}_{q_\theta,G}(g \mid x_i)$. Sometimes that evaluation can be done closed form (Macready and Wolpert, 2005, 2004). In blackbox optimization though, this is not possible.

Typically, when we cannot evaluate the terms $\mathbb{E}_{q_\theta,G}(g \mid x_i)$ in closed-form we use MC to estimate them. Since we have a product distribution, we can generate samples of the joint distribution $q_\theta(x)$ by sampling each of the marginals $q_i(x_i)$ separately. One can use those sample x's as queries to the oracle. Then, by appropriately averaging the oracle's responses to those queries, one can estimate each term $\mathbb{E}_{q_\theta,G}(g \mid x_i)$ (Wolpert and Bieniawski, 2004). The product factorization implies that our iterative procedure can be performed in a completely decentralized manner, with a separate program controlling each component x_i, and communicating *only* with the oracle[8].

In this scheme, once q_θ is modified, samples of the oracle that were generated from preceding q_θ's can no longer be directly used to estimate the terms $\mathbb{E}_{q_\theta,G}(g \mid x_i)$. However, there are several 'data-aging' heuristics one can employ to reuse such old data by down-weighting it.

In all these schemes, while we ultimately use Monte Carlo in the PC, it is delayed as long as possible in the course of solving (P5). This is the basis for calling this variant of PC 'delayed sampling'.

3.3 Advantages of PC

The PC transformation can substantially alter the optimization landscape. For a noise-free oracle $G(x)$, (P4) reduces to the problem of finding the x that minimizes $G(x)$. In contrast, (P5) is the problem of finding the θ that minimizes $\mathscr{F}_{\mathscr{G}}(\theta)$. The characteristics

6. This qp distance is just the free energy of q_θ for Hamiltonian function G and inverse temperature β_i. This gives a novel derivation of the physics injunction to minimize the free energy of a system.
7. One of these alternatives can be cast as a corrected version of the replicator dynamics of evolutionary game theory (Wolpert, 2004b). This may have interesting implications for GAs, which presume evolutionary processes.
8. Each such program may be thought of as an 'agent' who updates his probability distribution, and this 'collective' of agents performs optimization in a decentralized manner. This led to the name 'Probability Collective'.

of the problems of minimizing $G(x)$ and minimizing $\mathscr{F}_{\mathscr{G}}(\theta)$ can be vastly different. For example, suppose q_θ is log-concave in its parameters, and $\mathscr{F}_{\mathscr{G}}$ is pq KL distance. In this case, regardless of the function $G(x)$, $\mathscr{F}_{\mathscr{G}}(\theta)$ is a convex function of θ, over $\mathscr{Q}$. So the PC transformation converts a problem with potentially many local minima into a problem with none. See Wolpert et al. (2006) for a discussion of the geometry of the surface $\mathscr{F}_{\mathscr{G}}(\theta) : \theta \to \mathbb{R}$.

Since it works directly on distributions, PC can handle arbitrary data types. X can be categorical, real-valued, integer-valued, or a mixture of all of these, but in each case, the distribution over X is parametrized by a vector of real numbers. This means that all such problems 'look the same' to much of the mathematics of PC. Moreover, PC can exploit extremely well-understood techniques (like gradient descent) for optimization of continuous functions of real-valued vectors, and apply them to problems in these arbitrary spaces.

Optimizing over distributions can give sensitivity information: The distribution q_θ produced in PC will typically be tightly peaked along certain directions, while being relatively flat along other directions. This tells us the relative importance of getting the value of x along those different directions precisely correct.

We can set the initial distribution for PC to be a sum of broad peaks, each centered on a solution produced by some other optimization algorithm. Then, as that initial distribution gets updated in the PC algorithm, the set of solutions provided by those other optimization algorithms are in essence combined, to produce a solution that should be superior to any of them individually.

Yet another advantage to optimizing a distribution is that a distribution can easily provide multiple solutions to the optimization problem, potentially far apart in X. Those solutions can then be compared by the analyst in a qualitative fashion.

As discussed later, there are other advantages that accrue specifically if one uses the immediate-sampling variant of PC. These include the ability to reuse all old data, the ability to exploit prior knowledge concerning the oracle, and the ability to leverage PL techniques. See Wolpert et al. (2006) for a discussion of other advantages of PC, in particular in the context of distributed control.

3.4 Relation to Other Work

PC is related to several other optimization techniques. Consider, for instance, Response Surface Models (RSM)s Jones et al. (1998). In these techniques, one uses Design of Experiments (DOE) to evaluate the objective function at a set of points. Then, a low-order parametric function, often a quadratic, is fitted to these function values. Optimization of this 'response surface' or 'surrogate model' is considered trivial compared to the original optimization. The result of this surrogate optimization is then used to get more samples of the true objective at a different set of points. This procedure is then iterated using some heuristics, often in conjunction with trust-region methods to ensure validity of the low-order approximation. We note the similarities with PC in Table 2.

As another example, some variants of PC exploit MC techniques as discussed above, and thus stochastically generate populations of samples. In their use of random populations these variants of PC are similar to simulated annealing (Kirkpatrick et al., 1983), and even more so to techniques like EDA's Larraaga and Lozano (2001); De Bonet et al. (1997); Lozano et al. (2005) and the CE method (Rubinstein and Kroese, 2004). However, these other approaches do not explicitly pursue the optimization of the underlying distribution q_θ, as in (P5). Accordingly, those approaches cannot exploit situations in which (P5) can be solved without using a stochastically generated population (Macready and Wolpert, 2005, 2004). See Macready and Wolpert (2005) for a more extensive discussion of the relation of PC to other techniques.

RSM	PC
Fit parametric function to objective function values	Fit parametric distribution to target distribution
Heuristics to grow trust region	Cross-validation for regularization
DOE for sample points	Random sampling for sample points
Axis alignment of stencil matters	Parametrization can address axis alignment
Surrogate minimization not always easy	Implicit, probabilistic 'minimization' of surrogate

Table 2: Relation to RSM.

4. Immediate sampling

This section introduces a new PC technique called immediate sampling, and cursorily compares it to delayed sampling. As we have just described, in delayed sampling, we use algebra for as long as possible in our solution of (P5). When closed-form expressions can no longer be evaluated, we resort to MC techniques. In **immediate sampling**, we form an MC sample immediately, rather than delaying it as long as possible. That sample gives us an approximation to our objective, $\mathscr{F}_{\mathscr{G}}(\theta)$ for all $\theta \in \mathscr{Q}$. We then search for the θ that optimizes that sample-based approximate objective.

4.1 The General Immediate-sampling Algorithm

We begin with an illustrative example. Consider an integral transform $\mathscr{F}_{\mathscr{G}}(q_\theta)$, and use importance sampling to rewrite it as

$$\int dx\ h^1(x)\frac{\int dg\ P(g \mid x,\mathscr{G})F(g,q_\theta(x))}{h^1(x)} = \int dxdg\ h^1(x)P(g \mid x,\mathscr{G})\frac{F(g,q_\theta(x))}{h^1(x)}, \quad (12)$$

$$\triangleq \int dx\ h^1(x) r_{P(g|x,\mathscr{G}),h^1}(\theta). \quad (13)$$

where we call $h^1(x)$ the **sampling distribution**. Note that the r in Eq. 13 differs from the one defined in Eq. 10, as indicated by the extra subscript. This new r is used in the next section.

We form a **sample set** of N pairs $D^1 \equiv \{x^i, g^i\}$ by IID sampling the distribution $h^1(x)P(g \mid x,\mathscr{G})$ in the integrand of Eq. 12. That sampling is the 'immediate' Monte Carlo process. D^1 is equivalent to a set of N **sample functions**

$$r^i_{h^1}(x^i,\theta) \triangleq \frac{F(g^i,q_\theta(x^i))}{h^1(x^i)} \ :\ i = 1,\ldots N.$$

In the simplest version of immediate sampling, we would now use the functions $r^i_{h^1}(x^i,\theta)$, together with our prior knowledge (if any), to estimate the θ that minimizes $\mathscr{F}_{\mathscr{G}}(q_\theta)$. As an example, not using any prior knowledge, we could estimate $\mathscr{F}_{\mathscr{G}}(q_\theta)$ for any θ as

$$\int dx\ h^1(x) r_{P(g|x,\mathscr{G}),h^1}(\theta) \approx \frac{\sum_i r^i_{h^1}(x^i,\theta)}{N}. \quad (14)$$

This estimate is both an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$ and the maximum likelihood estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$. Moreover, it has these attributes for all θ. (This is the advantage of estimating $\mathscr{F}_{\mathscr{G}}(q_\theta)$ using importance sampling with a proposal distribution h that doesn't vary with

θ.) Accordingly, to estimate the θ that minimizes $\mathscr{F}_{\mathscr{G}}(q_\theta)$ we could simply search for the θ that minimizes $\sum_i r^i_{h^1}(x^i, \theta)$.[9]

Once again, even though the average in Eq. 14 is an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$ for any fixed θ, its minimizer is not an unbiased estimate of $\min_\theta \mathscr{F}_{\mathscr{G}}(q_\theta)$. This is because searching for the minimizing θ introduces bias. Therefore, one should use some other technique than directly minimizing the righthand side of Eq. 14 to estimate $\text{argmin}_\theta \mathscr{F}_{\mathscr{G}}(q_\theta)$.

4.2 Immediate Sampling with Multiple Sample Sets

In general, we will not end the algorithm after forming a single sample set D^1. Instead we will use a map η that takes D^1 to a new sampling distribution, h^2. We then generate new (x, g) pairs using h^2, giving us a new sample set D^2. We then iterate this process until we decide to end the algorithm, at which point we use all our samples sets together to estimate $\text{argmin}_\theta \mathscr{F}_{\mathscr{G}}(q_\theta)$.

To illustrate this we first present an example of a θ-estimation procedure we could run at the end of the immediate sampling algorithm. This example is just the extension of the maximum likelihood θ-estimation procedure introduced above to accommodate multiple sample sets. Let N be the total number of samples, drawn from M sample sets, with N_j samples in the j'th sample set. Let h^j be sample distribution for the j'th sample set, and $r^{i,j}$ the sample function value for the i'th element of the j'th sample set. Also define $D^j \equiv \{x^{i,j}, g^{i,j} : i = 1, \ldots N_j\}$. Then $\sum_{i=1}^{N_j} r^{i,j}_{h^j}(x^{i,j}, \theta)/N_j$ is an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$ for any sample set j. Accordingly, any weighted average of these estimates is an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(q_\theta)$:

$$\mathscr{F}_{\mathscr{G}}(q_\theta) \quad \approx \quad \sum_{j=1}^{M} w_j \sum_{i=1}^{N_j} \frac{r^{i,j}_{h^j}(x^{i,j}, \theta)}{N_j}. \tag{15}$$

Modulo unbiasedness concerns, we could then use the minimizer of Eq.15 as our estimate of $\text{argmin}_\theta \mathscr{F}_{\mathscr{G}}(q_\theta)$.

Say we have fixed on some such θ-estimation procedure to run at the end of the algorithm. The final step of each iteration of immediate sampling is to run η, the map taking the samples generated so far to a new h. Ideally, we want to use the η that, when repeatedly run during the algorithm, maximizes the expected accuracy of the final θ-estimation. However even for a simple θ-estimation procedure, determining this optimal η can be quite difficult. As discussed later, it is identical to the active learning problem in machine learning.

In this paper we adopt a two-step heuristic for setting η. In the first step, at the end of each iteration, we estimate the optimal q_θ based on all the sample sets generated so far, using Eq. 15. In the second step, we complete η by setting the new h to the current estimate of the optimal q_θ. At that point, the new h is used to generate a new sample set, and the process repeats.

4.3 Immediate Sampling with MCMC

For certain types of $\mathscr{F}_{\mathscr{G}}$, it is possible to form samples using other sampling methods like Markov Chain Monte Carlo (MCMC) (see Mackay, 2003; Bernardo and Smith, 2000;

9. Since q_θ is normalized, so is $\int dx\ F(G(x), q_\theta(x)) = \frac{\int dx\ F(G(x), q_\theta(x))}{\int dx\ q_\theta(x)}$. In Eq. 14 we fix the denominator integral to 1. In practice though, it may make sense to replace both of the integrals in this ratio with importance sample estimates of them. That means dividing the sum in Eq. 15 by $\sum_i q(x^i)/h(x^i)$ and then finding the θ that optimizes that ratio of sums, rather than the θ that just optimizes the numerator term (see Robert and Casella, 2004). For example, this can be helpful when one uses cross-validation to set β, as described below.

Berger, 1985). For example, if $\mathscr{F}_{\mathscr{G}}(\theta)$ is pq distance from the Boltzmann distribution p^β to q_θ, then we can use MCMC to form a sample set of p^β (not of q_θ). We can then use that sample set to form an unbiased estimate of $\mathscr{F}_{\mathscr{G}}(\theta)$ for any θ. But if β were to change, these old samples cannot be used directly. One would have to resort to additional techniques like rejection sampling in order to reuse these samples. The advantage of using importance sampling is that all previous samples can be reused by the simple expedient of modifying their likelihood ratios. Therefore, in this paper, we only consider sampling distributions h that can be sampled directly, without any need for techniques like MCMC.

4.4 Advantages of Immediate-Sampling PC

In contrast to delayed sampling, immediate sampling usually presents no difficulty with reusing old data, as shown above; all $(x^{i,j}, g^{i,j})$ pairs can be used directly. Note that we can also reuse data that was generated when F was different, for instance, data generated under a differerent β^i during a KL distance minimization procedure. As long as we store $h^j(x^{i,j})$ in addition to $g^{i,j}$ and $x^{i,j}$ for every sample, we can always evaluate $r^{i,j}_{h^j}(x^{i,j}, \theta)$ for any F.

Indeed, we can even comment on optimal ways of reusing this old data. Since each $r^{i,j}_{h^j}(x^{i,j}, \theta)$ is an unbiased estimate of the integral $\mathscr{F}_{\mathscr{G}}(\theta)$, any weighted average of the $r^{i,j}_{h^j}(x^{i,j}, \theta)$'s is also an unbiased estimate. This can be exploited in the θ-estimation procedure. For instance, consider the estimator of Eq. 15. If we have good estimates of the variances of the individual $r^{i,j}_{h^j}(x^{i,j}, \theta)$, we can weight the terms $r^{i,j}_{h^j}(x^{i,j}, \theta)$ to minimize the variance of the associated weighted average estimator. Those weights are proportional to the inverses of the variances (see Macready and Wolpert, 2005; Lepage, 1978, 1980). As discussed in Sec. 2.3, the accuracy of the associated MCO algorithm could be expected to improve under such weighting.

We can also shed light on how to go about gathering new data. As in the VEGAS Algorithm (Lepage, 1978, 1980), one could incorporate bias-variance considerations into the operator η that sets the next sampling distribution. To give an example, let Ξ be the range of η, and fix θ. Given Ξ and θ, one can ask what proposal distribution $h \in \Xi$ would minimize the sample variance of the estimator in Eq. 14. Intuitively, this is akin to asking how best to do active learning. In general, the answer to this question, the optimal sampling distribution $h(x)$, will be set by the function $r_{P(g|x,\mathscr{G}),h}(\theta)$, viewed as mapping $X \to \mathbb{R}$. Accordingly, for any fixed θ, one can use the MC samples generated so far to estimate the x-dependence of $r_{P(g|x,\mathscr{G}),h}(\theta)$, and thereby estimate the optimal $h \in \Xi$. One then uses that estimate as the next sampling distribution h.

Another advantage of immediate sampling over delayed sampling is that the analysis in delayed sampling relies crucially on the parametrization of the q's; some such parametrizations will permit the closed-form calculations of delayed sampling, and others will not. In immediate sampling, this problem disappears.

4.5 Implications of the Identity Between MCO and PL

For the case where $\mathscr{F}_{\mathscr{G}}(\theta)$ is an integral transform like Eq. 9, the PC optimization problem (P5) becomes a special case of minimizing a parametrized integral, the problem (P2). Formally, the equivalence is made by equating x with the parameter ϕ, g with w, and $g \times P(g \mid x, \mathscr{G})$ with $U(w, \phi)$. In particular, immediate sampling is a special case of MCO. This identity means that we can exploit the extremely well-researched field of PL to improve many aspects of immediate sampling. In particular:

- PL techniques like boosting (Schapire and Singer, 1999) and bagging (Breiman, 1994) can be used in (re)using old samples before forming new ones.

- Variants of active learning[10] can be used to set and update h. Some aspect of this are discussed in Sec. 6 below.
- Cross-validation is directly applicable in many ways: In our context, the curse of dimensionality arises if $\mathscr{Q}$ is very large. We can address this the conventional PL way, by adding a regularization function of q_θ to the objective function. The parameters controlling this regularization can be updated dynamically, as new data is generated, using cross-validation

  To use cross-validation this way, one forms multiple partitions of the current data. For each such partition, one calculates the optimal q_θ for the training subset of that partition. One then examines error on the validation subset of that partition. More precisely, one calculates the *unregularized* objective value on the held-out data.
- More generally, we can use cross-validation to dynamically update *any* parameters of the immediate sampling algorithm. For example, we can update the 'temperature' parameter β of the Boltzmann distribution, arising in both qp and pq KL distance, this way.

  Note that doing this does not involve making more calls to the oracle.
- We can also use cross-validation to choose the best model class (parametrization) for q_θ, among several candidates.
- As an alternative to all these uses of cross-validation, one can use stacking to dynamically combine differrent temperatures, different parametrized density functions, and so on.
- One may also be able to apply kernel methods to do the density estimation (see Macready, 2005).

5. Experiments

In this section, we demonstrate the application of PL and immediate-sampling PC techniques to the unconstrained optimization of continuous functions, both deterministic and nondeterministic. We first describe our choice of $\mathscr{F}_\mathscr{Q}$, in this case pq KL distance. Next, as an illustrative example, we apply immediate sampling to the simplest of optimization problems, where the objective is a 2-D quadratic. Subsequently, we apply it to deterministic and stochastic versions of two well-known unconstrained optimization benchmarks, the Rosenbrock function and the Woods function.

We highlight the use of PL techniques to enhance optimizer performance on these benchmark problems. In particular, we show that cross-validation for regularization yields a performance improvement of an order of magnitude. We then show that cross-validation for model-selection results in improved performance, especially in the early stages of the algorithm. We also show that bagging can yield significant improvements in performance.

5.1 Minimizing pq KL Distance

Recall that the integral form of pq KL distance is

$$\mathrm{KL}(p\|q) = \int dx\, p(x) \ln\left(\frac{p(x)}{q(x)}\right).$$

It is easy to show that when there are no restrictions on q being a parametrized density, pq KL distance is minimized if $p = q$. However, owing to sampling considerations, we

10. Active learning in the precise machine learning sense uses current data to decide on a new query x to feed to the oracle. We use the term more loosely here, to refer to any scheme for using current data to dynamically modify a process for generating for future queries.

usually choose q to be some parametric distribution q_θ. In this case, we want to find the parameter vector θ that minimizes $\mathrm{KL}(p\|q_\theta)$. Since the target distribution p is derived purely from $\mathscr{G}$ and is independent of q_θ, minimizing pq KL distance is equivalent to the following cross-entropy minimization problem.

$$\begin{array}{ll} \text{minimize} & -\int dx\, p(x) \ln\left(q(x)\right), \\ \text{subject to} & \int dx\, q(x) = 1, \\ & q(x) \geq 0 \quad \forall x. \end{array} \tag{16}$$

5.1.1 Gaussian Densities

If q is log-concave in its parameters θ, the minimization problem (16) is a convex optimization problem. In particular, consider the case where $X = \mathbb{R}^n$, and q_θ is a multivariate Gaussian density, with mean μ and covariance Σ, parametrized as follows,

$$q_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right),$$

then the optimal parameters are given by matching first and second moments of p and q_θ.

$$\begin{array}{rcl} \mu^\star & = & \int dx\, x\, p(x), \\ \Sigma^\star & = & \int dx\, (x-\mu^\star)(x-\mu^\star)^T p(x). \end{array}$$

It is easy to generalize this to the case where $X \subset \mathbb{R}^n$, by making a suitable modification to the definition of p. This is described in Sec. 5.2.1.

5.1.2 Immediate Sampling with a Single Gaussian

Using importance sampling, we can convert the cross-entropy integral in Eq. 16 to a sum over data points, as follows.

$$\frac{1}{N}\sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i)\right),$$

where D is the data set $\{(x^i, g^i)\}, i = 1, \ldots, N$. This sets up the minimization problem for immediate sampling for pq KL distance.

$$\text{minimize } -\sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i)\right). \tag{17}$$

Denote the likelihood ratios by $s^i = p(x^i)/h(x^i)$. Differentiating Eq. 17 with respect to the parameters μ and Σ^{-1} and setting them to zero yields[11]

$$\begin{array}{rcl} \mu^\star & = & \dfrac{\sum_D s^i x^i}{\sum_D s^i} \\ \Sigma^\star & = & \dfrac{\sum_D s^i (x^i-\mu^\star)(x^i-\mu^\star)^T}{\sum_D s^i} \end{array}$$

11. **Remarks:**
    1. As expected, these formulæ, in the infinite-data limit, are identical to the moment-matching results for the full-blown integral case.
    2. The formulæ resemble those for MAP density estimation, often used in supervised learning to find the MAP parameters of a distribution from a set of samples. The difference in this case is that each sample point is weighted by the likelihood ratio s^i, and is equivalent to 'converting' samples from h to samples from p.

5.1.3 Mixture Models

The single Gaussian is a fairly restrictive class of models. Mixture models can significantly improve flexibility, but at the cost of convexity of the KL distance minimization problem. However, a plethora of techniques from supervized learning, in particular the Expectation Maximization (EM) algorithm, can be applied with minor modifications.

Suppose q_θ is a mixture of M Gaussians, that is, $\theta = (\mu, \Sigma, \phi)$ where ϕ is the mixing p.m.f, we can view the problem as one where a hidden variable z decides which mixture component each sample is drawn from. We then have the optimization problem

$$\text{minimize } -\sum_D \frac{p(x^i)}{h(x^i)} \ln\left(q_\theta(x^i, z^i)\right).$$

Following the standard EM procedure, we multiply and divide the quantity inside the logarithm by some $Q_i(z^i)$, where Q_i is a distribution over the possible values of z^i. As before, let s^i be the likelihood ratio of the i'th sample.

$$\text{minimize } -\sum_D s^i \ln\left(\sum_{z^i} Q_i(z^i) \frac{q_\theta(x^i, z^i)}{Q_i(z^i)}\right)$$

Then using Jensen's inequality, we can take Q_i outside the logarithm to get a lower bound. To make this lower bound tight, choose $Q_i(z^i)$ to be the constant $p(z^i|x^i)$. Finally, differentiating with respect to μ_j, Σ_j^{-1} and ϕ_j gives us the EM-like algorithm:

$$\begin{array}{lrcl}
\text{E-step: For each i, set} & Q_i(z^i) & = & p(z^i|x^i), \\
\text{that is,} & w_j^i & = & q_{\mu,\Sigma,\phi}(z^i = j|x^i), \quad j = 1, \ldots, M. \\
\text{M-step: Set} & \mu_j & = & \dfrac{\sum_D w_j^i s^i \, x^i}{\sum_D w_j^i s^i}, \\
& \Sigma_j & = & \dfrac{\sum_D w_j^i s^i \, (x^i - \mu_j)(x^i - \mu_j)^T}{\sum_D w_j^i s^i}, \\
& \phi_j & = & \dfrac{\sum_D w_j^i s^i}{\sum_D s^i},
\end{array}$$

Since this is a nonconvex problem, one typically runs the algorithm multiple times with random initializations of the parameters.

5.2 Implementation Details

In this section we describe the implementation details of an iterative immediate-sampling PC algorithm that uses the Gaussian mixture models described in the previous section to minimize pq KL distance to a Boltzmann target parametrized by β. We also describe the modification of a variety of techniques from parmetric learning that significantly improve performance of this algorithm. An overview of the procedure is presented in Algorithm 1.

5.2.1 Example: Quadratic $G(x)$

Consider the 2-D box $X = \{x \in \mathbb{R}^2 \mid \|x\|_\infty < 1\}$. Consider a simple quadratic on X,

$$G_Q(x) = x_1^2 + x_2^2 + x_1 x_2, \quad x \in X.$$

The surface and contours of this simple quadratic on X are shown in Fig. 1. Also shown are the corresponding Boltzmann target distributions p^β on X, for $\beta = 2, 10$ and 50. As

Algorithm 1 Overview of pq minimization using Gaussian mixtures

1: Draw uniform random samples on X
2: Initialize regularization parameter β
3: Compute $G(x)$ values for those samples
4: **repeat**
5: Find a mixture distribution q_θ to minimize sampled pq KL distance
6: Sample from q_θ
7: Compute $G(x)$ for those samples
8: Update β
9: **until** Termination
10: Sample final q_θ to get solution(s).

can be seen, as β increases, p^β places increasing probability mass near the optimum of $G(x)$, leading to progressively lower $\mathbb{E}_{p^\beta} G(x)$. Also note that since $G(x)$ is a quadratic, $p^\beta(x) \propto \exp(-\beta G(x))$ is a Gaussian, restricted to X and renormalized. We now 'fit' a Gaussian density q_θ to the Boltzmann p^β by minimizing $\mathrm{KL}(p^\beta \| q_\theta)$, for a sequence of increasing values of β. Note that q_θ is a distribution over $\mathbb{R}^2$, and G_Q is not defined everywhere in $\mathbb{R}^2$. Therefore, we extend the definition of G_Q to all of $\mathbb{R}^2$ as follows.

$$G_Q(x) = \begin{cases} x_1^2 + x_2^2 + x_1 x_2, & x \in X. \\ \infty & \text{otherwise.} \end{cases}$$

Now $p^\beta = 0$ for all $x \notin X$, and the integral for KL distance can be reduced to an integral over X. This means that samples outside X are not considered in our computations.

5.2.2 Constant β

First, we fix $\beta = 5$, and run a few iterations of the PC algorithm. To start with, we draw $N_j = 30$ samples from the uniform distribution on X. The best-fit Gaussian is computed using the immediate sampling procedure outlined in the preceding section. At each successive iteration, $N_j = 30$ more samples are drawn from the current q_θ and the algorithm proceeds. A total of 6 such iterations are performed. The 90% confidence ellipsoids corresponding to p^β (heavy line) and the iterates of q_θ (thin line) are shown in Fig. 2. Also shown are the corresponding values of $\mathbb{E}_{q_\theta} G(x)$, computed using the sample mean of $G_Q(x)$ for 1000 samples of x drawn from each q_θ, and $\mathrm{KL}(p^\beta \| q_\theta)$, computed as the sample mean of $\ln(p^\beta(x)/q_\theta(x))$ for 1000 samples of x drawn according to p^β.

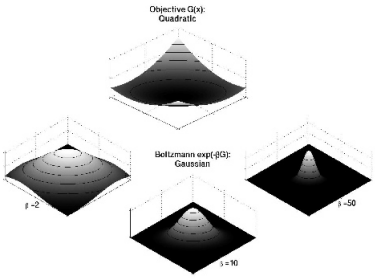
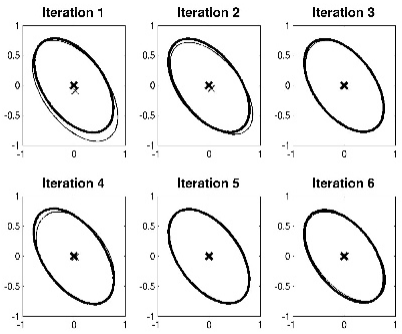


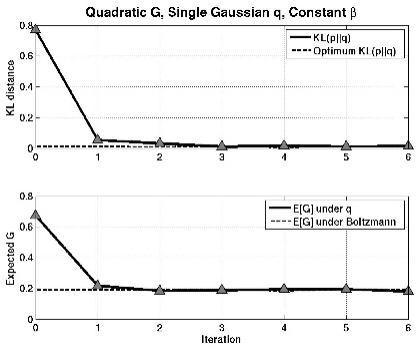
Figure 1: Quadratic $G(x)$ and associated Gaussian targets

5.2.3 Varying β

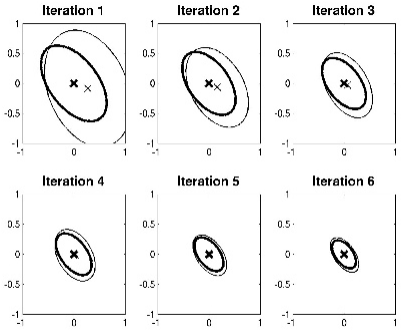
Next, we change β between iterations, in the 'update β' step shown in algorithm(1). With the same algorithm parameters, we start with $\beta = 10$, and at each iteration, we use a multiplicative update rule $\beta \leftarrow k_\beta \beta$, for some constant $k_\beta > 1$, in this case, 1.5. As the algorithm progresses, the increasing β causes the target density p^β to place increasing probability mass on regions with low $G(x)$, as shown in Fig. 1. Since the distributions q_θ



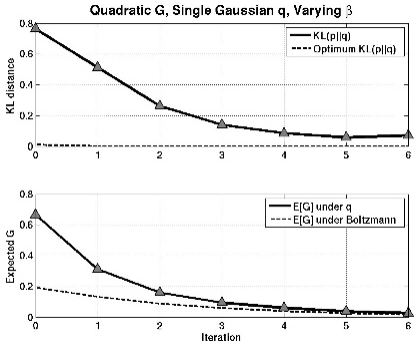
(a) Constant β: Confidence ellipsoids



(b) Constant β: KL distance and expected G



(c) Varying β: Confidence ellipsoids



(d) Varying β: KL distance and expected G

Figure 2: PC iterations for quadratic $G(x)$.

are best-fits to p, successive iterations will generate lower $\mathbb{E}_{q_\theta} G(x)$. The 90% confidence ellipsoids and evolution of $\mathbb{E}_{q_\theta} G(x)$ and KL distance are shown in Fig. 2.

5.2.4 Cross-validation to Schedule β

In more complex problems, it may be difficult to find a good value for the β update ratio k_β. However, we note that the objective $\mathrm{KL}(p^\beta \| q_\theta)$ can be viewed as a regularized version of the original objective, $\mathbb{E}_{q_\theta}[G(x)]$. Therefore, we use the standard PL technique of cross-validation to pick the regularization parameter β from some set $\{\beta\}$. At each iteration, we partition the data set D into training and test data sets D_T and D_V. Then, for each $\beta \in \{\beta\}$, we find the optimal parameters $\theta^\star(\beta)$ using only the training data D_T. Next, we test the associated $q_{\theta^\star(\beta)}$ on the test data D_V using the following performance measure.

$$\widehat{g}(\theta) = \frac{\sum_{D_V} \frac{q_\theta(x^i) G(x^i)}{h(x^i)}}{\sum_{D_V} \frac{q_\theta(x^i)}{h(x^i)}}, \tag{18}$$

The objective $\widehat{g}(\theta)$ is an estimate[12] of the unregularized objective $\mathbb{E}_{q_\theta}[G(x)]$. Finally, we set $\beta^\star = \arg\min_{\beta\in\{\beta\}} \widehat{g}(\theta^\star(\beta))$, and compute $\theta^\star(\beta^\star)$ using *all* the data D. Note that the whole cross-validation procedure is carried out without any more calls to the oracle $\mathscr{G}$.

We demonstrate the functioning of cross-validation on the well-known Rosenbrock problem in two dimensions, given by

$$G_R(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

over the region $X = \{x \in \mathbb{R}^2 \mid \|x\|_\infty < 4\}$. The optimum value of 0 is achieved at $x = (1, 1)$. The details of the cross-validation algorithm used are presented in Algorithm 2. For this

Algorithm 2 Cross-validation for β.

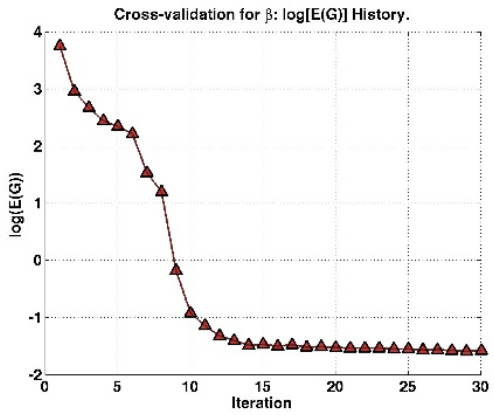
Initialize interval extension count `extIter` $= 0$, and `maxExtIter` and β_0..
repeat
  At $\beta = \beta_0$, consider the interval $\Delta\beta = [k_1\beta_0, k_2\beta_0]$.
  Choose $\{\beta\}$ be a set of n_β equally-spaced points in $\Delta\beta$.
  Partition the data into K random disjoint subsets.
  for each fold k, **do**
    Training data is the union of all but the k^{th} data partitions.
    Test data is the k^{th} partition.
    for β_i in $\{\beta\}$, **do**
      Use training data to compute optimal parameters $\theta^\star(\beta_i, D_{T_k})$.
      Use test data to compute held-out performance $\widehat{g}(\theta^\star(\beta_i, D_{V_k}))$, from Eq. 18.
    end for
  end for
  Compute average held-out performance, $\overline{g}(\beta_i)$, of $\widehat{g}(\theta^\star(\beta_i, D_{V_k}))$.
  Fit a quadratic $Q(\beta)$ in a least-squares sense to the data $(\beta_i, \overline{g}(\beta_i))$.
  if Q is convex **then**
    Set optimum regularization parameter $\beta^\star = \arg\min_{\beta\in\Delta\beta} Q(\beta)$.
  else
    Fit a line $L(\beta)$ in a least-squares sense to the data $(\beta_i, \overline{g}(\beta_i))$.
    Choose $\beta^\star = \arg\min_{\beta\in\Delta\beta} L(\beta)$.
  end if
  Increment `extIter`
  Update $\beta_0 \leftarrow \beta^\star$
until `extIter` $>$ `maxExtIter` or Q is convex.

experiment, we choose

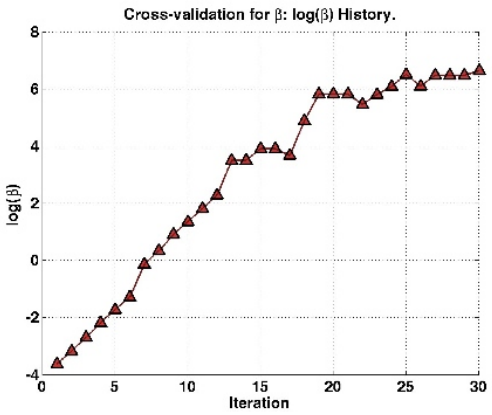
$$\begin{array}{rrrrrr} \texttt{maxExtIter} = & 4, & k_1 = & 0.5, & k_2 = & 2, \\ N_j = & 10, & n_\beta = & 5, & K = & 10. \end{array}$$

The histories of $\mathbb{E}_q G(x)$ and β are shown in Fig. 3. Also shown are plots of the fitted $Q(\beta)$ at iterations 8 and 15. As can be seen, the value of β sometimes *decreases* from one

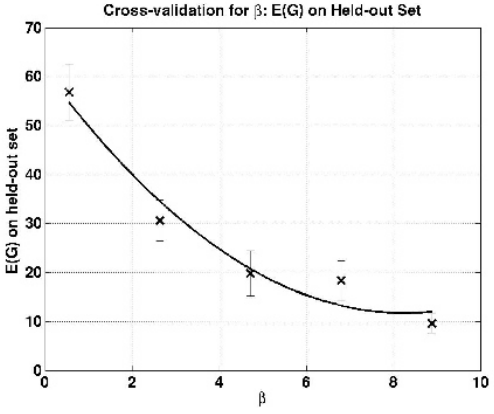
12. The reason for dividing by the sum of $q(x^i)/h(x^i)$ is as follows. If the training data is such that no probability mass is placed on the test data, the numerator of $\widehat{g}_{q_\theta}$ is 0, regardless of the parameters of q_θ. In order to avoid this peculiar problem, we divide by the sum of $q(x^i)/h(x^i)$, as desribed by Robert and Casella (2004).



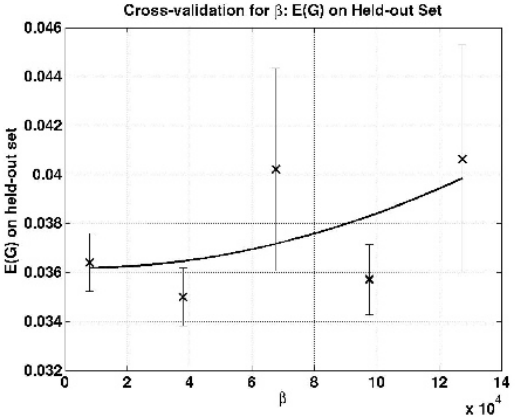
(a) $\mathbb{E}_q G(x)$ history.



(b) β history.



(c) Fit $Q(\beta)$, iteration 8.



(d) Fit $Q(\beta)$, iteration 15.

Figure 3: Cross-validation for β: 2-D Rosenbrock $G(x)$.

iteration to the next, which can never happen in any fixed multiplicative update scheme.

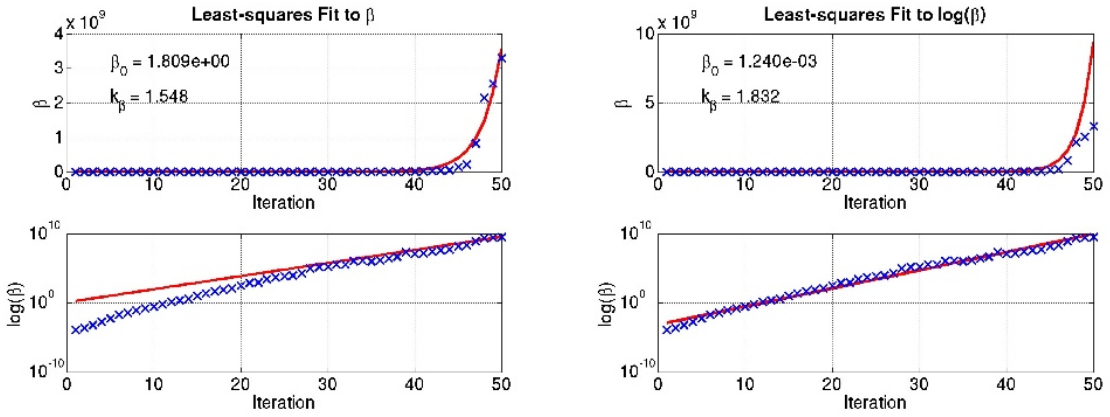
We now demonstrate the need for an automated regularization scheme, on another well-known test problem in $\mathbb{R}^4$, the Woods problem, given by

$$\begin{aligned} G_{\text{woods}}(x) &= 100(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10.1[(1 - x_2)^2 + (1 - x_4)^2] + 19.8(1 - x_2)(1 - x_4). \end{aligned}$$

The optimum value of 0 is achieved at $x = (1, 1, 1, 1)$. We run the PC algorithm 50 times with cross-validation for regularization. For this experiment, we used a single Gaussian q, and set

$$\begin{array}{rrrrrr} \texttt{maxExtIter} = & 4, & k_1 = & 0.5, & k_2 = & 3, \\ N_j = & 20, & n_\beta = & 5, & K = & 10. \end{array}$$

From these results, we then attempt to find the best-fit multiplicative update rule for β, only to find that the average β variation is not at all well-approximated by *any* fixed update $\beta \leftarrow k_\beta \beta$. This poor fit is shown in Fig. 4, where we show a least-squares fit to both β and $\log(\beta)$. In the fit to $\log(\beta)$ the final β is off by over 100%, and in the fit to

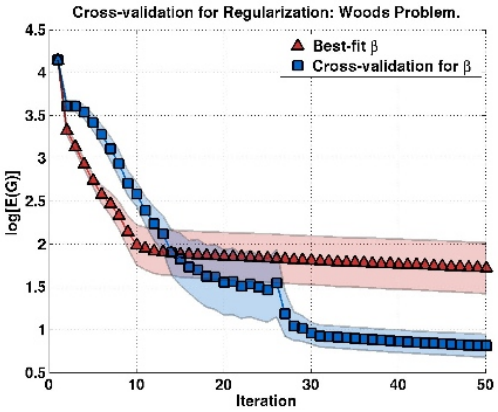


(a) Least-squares fit to β:

(b) Least-squares fit to $\log(\beta)$:

Figure 4: Best-fit β update rule.

β, the initial β is off by several orders of magnitude. We then compare the performance of cross-validation to that of PC algorithm using the fixed update rule derived from the best least-squares fit to $\log(\beta)$. From a comparison over 50 runs, we see that using this best-fit update rule performs extremely poorly - cross-validation yields an improvement in final $\mathbb{E}_{q_\theta} G(x)$ by over an order of magnitude, as shown in Fig. 5.



(a) $\log(\mathbb{E}_q G)$ history.

Figure 5: Cross-validation beats best-fit fixed β update: 4-D Woods $G(x)$.

5.2.5 Bagging

While regularization is a method to decrease bias, bagging is a well-known variance-reducing technique. Bagging is easily incorporated in our algorithm. Suppose, at some stage in the algorithm, we have N samples (x^i, g^i), we resample our existing data set exactly N times with replacement. This gives us a different set of data set D', which also contains some duplicates. We compute optimal parameters $\theta^\star(D')$. We repeat this resampling process k_b times and uniformly average the resulting optimal densities $q_{\theta^\star(D'_k)}, k = 1, \ldots, k_b$.

We demonstrate this procedure, using the Rosenbrock function and a single Gaussian q_θ. In this experiment, we also demonstrate the ability of PC to handle non-deterministic oracles by adding uniform random noise to every function evaluation, that is, $(g \mid x, G) \sim \mathcal{U}[-0.25, 0.25]$. For this experiment, $N_j = 20, k_b = 5$. The β update is performed using the same cross-validation algorithm described above. Fig. 6 shows the results of 50 runs of the PC algorithm with and without bagging. We see that bagging finds better solutions, and

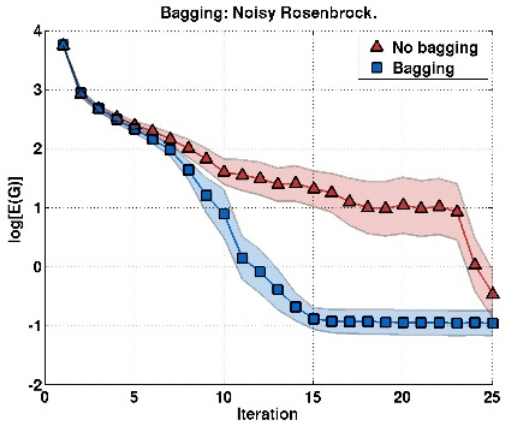


Figure 6: Bagging improves performance: Noisy 2-D Rosenbrock.

moreover, it reduces the variance between runs. Note that the way we use bagging, we are only assured of improved variance for a single MC estimation at a given θ, and not over the whole MCO process of searching over θ.

5.2.6 Cross-validation for Regularization and Model Selection

In many problems like the Rosenbrock, a single Gaussian is a poor fit to p^β for many values of β. In these cases, we can use a mixture of Gaussians to obtain a better fit to p^β. We now describe the use of cross-validation to pick the number of components in the mixture model. We use an algorithm very similar to the one described for regularization. In these experiments, we use a greedy algorithm to search over the joint space of β and models:

1. We first pick the regularization parameter β, using Algorithm 2.

2. For that β, we use Algorithm 3 to pick the number of mixture components.

For this experiment, the details are the same as the preceding section, but without bagging. The set of models $\{\{\phi\}\}$ is the set of Gaussian mixtures with one, two or three mixing components. Fig. 7 shows the variation of $\mathbb{E}_q(G)$ vs. iteration. The mixture model is much quicker to yield lower expected G, because the Boltzmann at many values of β is better approximated by a mixture of Gaussians. However, note that the mixture models performs poorly towards the end of the run. The reason for this is as follows: No shape regularization was used during the EM procedure. This means that the algorithm often samples from nearly degenerate Gaussians. These 'strange' sample sets hurt the subsequent performance of importance sampling, and hence of the associated MCO problem. This can be alleviated by using some form of shape regularization in the EM algorithm.

Algorithm 3 Cross-validation for model selection.

Initialize set $\{\{\phi\}\}$ of model classes $\{\phi\}$ to search over.
Partition the data into K disjoint subsets.
for each fold k, **do**
  Training data is all but the k^{th} data partitions.
  Test data is the k^{th} data partition.
  for $\{\phi_i\}$ in $\{\{\phi\}\}$ **do**
    Compute the optimal parameter set $\theta^\star(D_{T_k}) \in \{\phi_i\}$
    Compute held-out performance $\widehat{g}(\theta^\star(D_{V_k}))$
  end for
  Compute the sample held-out performance, $\overline{g}(\{\phi_i\})$, from Eq. 18.
end for
Choose best model class $\{\phi^\star\} = \arg\min_{\phi_i} \overline{g}(\{\phi_i\})$.

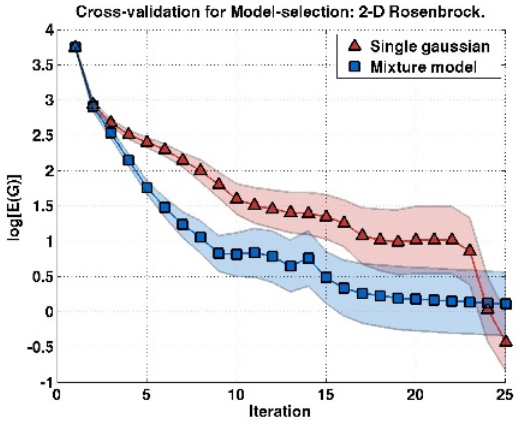


Figure 7: Cross-validation for regularization and model-selection: 2-D penalty function $G(x)$.

6. Fit-based Monte Carlo

Thus far, we have not exploited the locations of the samples in constructing esimates. In this section, we discuss the incorporation of sample locations to improve both MC and MCO.

6.1 Fit-based MC Estimation of Integrals

We first consider MC estimation of an integral, presented at the beginning of Sec. 2.3. Recall from that discussion, that to accord with MCO notation, we write the integral to be estimated as $\mathscr{L}(\phi) = \int dw\, U(w, \phi)$ for some fixed ϕ. In this notation the sampling of v provides a sample set $\{(w^i, U(w^i, \phi)) : i = 1, \ldots N)$. The associated sum $\hat{\mathscr{L}}_{\{(w^i, U(w^i,\phi))\}} \equiv \hat{\mathscr{L}}(\phi)$ then serves as our estimate of the integral $\mathscr{L} \equiv \mathscr{L}(\phi)$.

In forming the estimate $\hat{\mathscr{L}}_{\{(w^i, U(w^i,\phi))\}}$ we do not exploit the relationships between the locations of the sample points and the associated values of the integrand. Indeed, since

those locations $\{w^i\}$ do not appear directly in that estimate, $\hat{\mathscr{L}}_{\{(w^i,U(w^i,\phi))\}}$ is unchanged even if those locations changed in such a way that the values $\{r^i\}$ stayed the same.

The idea behind **fit-based** (FB) Monte Carlo is to leverage the location data to replace $\hat{\mathscr{L}}_{\{(w^i,U(w^i,\phi))\}}$ with a more accurate estimate of $\mathscr{L}$. The most straightforward FB MC treats the sample pairs $\{(w^i, U(w^i,\phi)) : i = 1, \ldots N)\}$ as a training set for a supervised-learning algorithm. Running such an algorithm produces a fit $\tilde{U}(.,\phi)$ taking w's into $\mathbb{R}$. This fit is an estimate of the actual oracle $U(,\phi)$, and this fit defines an estimate for the full integral,

$$\tilde{\mathscr{L}}_{\{w^i,U(w^i,\phi)\}} \equiv \int dw\, \tilde{U}(w,\phi) \tag{19}$$

We will sometimes omit the subscript and just write $\tilde{\mathscr{L}}(\phi)$ or even just $\tilde{\mathscr{L}}$. In this most straightforward version of FB MC, we use $\tilde{\mathscr{L}}$ as our estimate of $\mathscr{L}$ rather than $\hat{\mathscr{L}}$.

In some circumstances one can evaluate $\tilde{\mathscr{L}}$ in closed form. A recent paper reviewing some work on how to do this with Gaussian processes is given by Rasmussen and Gharamani (2003). In other circumstances one can form low-order approximations to $\mathscr{L}$, for example using Laplace approximations (see Robert and Casella, 2004). Alternatively, conventional deterministic grid-based approximation of the integral $\mathscr{L}$ can be cast as a degenerate version of closed-form fit-based estimation[13] of $\mathscr{L}$.

More generally, one can form an approximation to the integral $\tilde{\mathscr{L}}(\phi)$ by MC sampling of $\tilde{U}(.,\phi)$. Generating these **fictitious samples** of $\tilde{U}(.,\phi)$ does not incur the expense of calling the actual oracle $U(.,\phi)$. So, in this approach, we run MC twice. The first time, we generate the **factual samples** $\{(w^i, U(w^i,\phi)) : i = 1, \ldots N)\}$. Given those samples, we form the fit to them, $\tilde{U}(.,\phi)$. We then run a second MC process using the fictitious oracle $\tilde{U}(.,\phi)$.

Note that in all of these approaches, the original sampling distribution v does not directly arise, that is, the values $\{v(w^i)\}$ do not arise. In particular, if one were to change those values without changing the factual sample locations $\{w^i\}$, then the estimate $\tilde{\mathscr{L}}_{\{(w^i,U(w^i,\phi))\}}$ would not change. Of course, a different v would result in a different sample set, and thereby a different estimate, but *given a sample set*, the sampling distribution is immaterial. This is typically the case with FB MC estimators, and it contrasts with the estimator $\hat{\mathscr{L}}_{\{(w^i,U(w^i,\phi))\}}$, which would change if v were changed without changing the factual sample locations.

Note also that the factual samples underlying the fit $\tilde{U}(.,\phi)$ are exact samples of the factual oracle, $U(.,\phi)$. In contrast, since in general $\tilde{U}(.,\phi) \neq U(.,\phi)$, the fictitious samples will be erroneous, if viewed as samples of $U(.,\phi)$. Since we are ultimately concerned with an integral of $U(.,\phi)$, this suggests that the fictitious samples should be weighted less than the factual samples. This might be the case even if one had infinitely many fictitious samples. In fact, even if one could evaluate $\tilde{\mathscr{L}}$ in closed form, it might make sense not to use it directly as our final estimate of $\mathscr{L}$. Instead, combining it with the importance-sampling estimate $\hat{\mathscr{L}}$ might improve the estimate.

13. To see this, modify the MC process to be sampling without replacement. Choose the proposal distribution v for this process to be a sum of delta functions. The centers of those delta functions give the points on a regular grid of points in the space of allowed x's. Have the number of samples equal the number of such grid points. Finally, have our fit to the samples, $\tilde{U}(.,\phi)$, be a sum of step-wise constant functions, going through the sample points. The closed form integral of that fit given by Eq. 19 is just the Reimann approximation to the original integral, $\int dw\, U(w,\phi)$.

6.2 Bayesian Fit-based MCO

We now extend the discussion to MCO by allowing ϕ to vary. As with the case of FB MC estimation of an integral, the most straightforward version of FB MCO uses $\tilde{\mathscr{L}}(\phi)$ rather than $\hat{\mathscr{L}}(\phi)$ as our estimate of $\mathscr{L}(\phi)$. This means that we use $\text{argmin}_\phi[\tilde{\mathscr{L}}(\phi)]$ rather than $\text{argmin}_\phi[\hat{\mathscr{L}}(\phi)]$ as our estimate of the ϕ optimizing $\mathscr{L}(\phi)$.

$\hat{\mathscr{L}}(\phi)$ is a sum, whereas $\tilde{\mathscr{L}}(\phi)$ is an integral. This means that different algorithms are required to find the ϕ optimizing them. Indeed, optimizing $\tilde{\mathscr{L}}(\phi)$ is formally the same type of problem as optimizing $\mathscr{L}(\phi)$; both functions of ϕ are parametrized integrals over w. So if needed, we can use PLMCO techniques to optimize $\tilde{\mathscr{L}}(\phi)$. Again, as with MC, the integrand of $\tilde{\mathscr{L}}(\phi)$ is not the factual oracle. So, minimizing $\tilde{\mathscr{L}}(\phi)$ using PLMCO sampling techniques will not require making additional calls to the factual oracle.

Consider a Bayesian approach to forming the fit. Our problem is to solve the MCO problem (P2), given that the factual oracle U is not known, and we only can generate samples of U. Adopting a fully Bayesian perspective, since U is not known, we must treat it as a random variable. So we have a posterior distribution over all possible oracles, reflecting all the data we have concerning the factual oracle. We then use that posterior to try to solve (P2).

Say that in the usual way that our data contains the w's and the associated functions $U(w, .)$ of a sample set that was generated by importance sampling U (see Sec. 2). More generally, we may have additional data, for instance, the gradients of U at the sample points. For simplicity though, we restrict attention to the case where the provided information is only the sample set of functions, $\{w^i, r^i(.)\}$, together with v.

We will use D to refer to a sample set for MC or MCO, and for immediate sampling in particular. So we write our posterior over oracles[14] as $P(U_c \mid D)$. Using this notation, the goal in Bayesian FB MCO is to exploit $P(U_c \mid D)$ to improve our estimate of the ϕ that minimizes $\int dw\, U(w, \phi)$.

How should we use $P(U_c \mid D)$ to estimate the solution to (P2)? Bayesian decision theory tells us to minimize posterior expected loss, $\int dU_c\, P(U_c \mid D) L(\phi, U_c)$. Given the loss function of Eq. 4, that means we wish to solve

$$\text{(P6):}\ \min_\phi \int dU_c\, P(U_c \mid D) L(\phi, U_c) \quad = \quad \min_\phi \int dw_c dU_c\, P(U_c \mid D) U_c(w_c, \phi).$$

To avoid confusion, the variable of integration is written as w_c, to distinguish it from w's in the integral $\int dw\, U(w, \phi)$. The solution to (P6) is our best possible guess of the ϕ solving problem (P2), given the sample set D. Finding that solution is a problem of minimizing a parametrized integral.

Sometimes we may be able to solve (P6) in closed form, even when we cannot solve (P2) in closed form. Performing the integral over U_c may simplify the remaining integral over w_c. More generally, we can address (P6) using MCO techniques, and in particular using PLMCO.[15]

To solve (P6) with PLMCO one generates fictitious samples by sampling one distribution over U_c's and one over w_c's. This MC process does not involve calls to the actual oracle U, but samples a new distribution over U_c's, to generate counter-factual U_c's, and then samples those U_c's.

14. Practically, when running a computer experiment, U is the actual oracle generating D according to a likelihood $P(D \mid U)$. On the other hand, the posterior $P(U_c \mid D)$ reflects both that likelihood and a prior $P(U_c)$ assumed by the algorithm. So, U_c is a random variable, whereas U refers to the single true factual oracle.

15. To see that explicitly, rewrite the integral in (P2) as $\int dz\, V(z, \phi)$, and identify values of z with pairs (w_c, U_c), while taking $V(z, \phi) = V(w_c, U_c, \phi) = P(U_c \mid D) U_c(w_c, \phi)$.

6.3 Example: Fit-based Immediate Sampling

To illustrate the foregoing we consider the variant of MCO given by immediate sampling with a noise-free oracle. In the simple version of MCO considered just above, the estimate we make for ϕ has no effect on what points are chosen for any future calls we might make to the oracle. For simplicity, we restrict attention to the analogous formulation of immediate sampling. Using immediate sampling terminology, this means that we only consider the issue of how best to estimate θ after the immediate sampling algorithm has exhausted all its calls to the oracle. We do not consider the more general active learning issue, of how best to estimate θ when this estimate will be affect further calls to the oracle. However see Sec. 6.6 below.

Recall that in immediate sampling, identifying w_c with x_c and ϕ with θ, $U_c(w_c, \phi)$ becomes $F(G_c(x_c), \theta)$. Given a sample set D of $(x, G(x))$ pairs generated from a noise-free factual oracle, our Bayesian optimization problem in immediate sampling is to find the θ that minimizes

$$\begin{aligned} \mathbb{E}(\mathscr{F}_{G_c}(\theta) \mid D) &= \mathbb{E}_{P(G_c|D)}[\int dx_c F(G_c(x_c), q_\theta(x_c))] \\ &= \int dx_c dG_c \; P(G_c \mid D) F(G_c(x_c), q_\theta(x_c)) \\ &\triangleq \tilde{\mathscr{F}}_D(\theta). \end{aligned} \tag{20}$$

Contrast this with Eq. 11. In particular, the inner integral in Eq. 20 runs over fictitious oracles G_c that are generated according to $P(G_c \mid D)$, whereas in Eq. 11, G is the factual oracle.

In some circumstances one can evaluate the integral in Eq. 20 algebraically, to give a closed form function of θ. In other cases, we can algebraically evaluate an accurate low-order approximation, to again give a closed form function of θ. For the rest of this subsection, however, we consider the situation where neither of these possibilities hold.

To address this situation, we approximate the integral in Eq. 20 using importance sampling. However, to do this, we may have to importance-sample over two domains. The first sampling is over sample locations x_c, using some sampling distribution h_c. (As an example, we can simply choose $h_c = h$.) The second sampling is over possible oracles G_c, using some sampling distribution H. More precisely, write

$$\mathbb{E}(\mathscr{F}_{G_c}(\theta) \mid D) = \int dx_c \, dG_c \, \left[h_c(x_c) H(G_c)\right] \, \frac{P(G_c \mid D)}{h_c(x_c) H(G_c)} F(G_c(x_c), q_\theta(x_c)). \tag{21}$$

To approximate this integral generate N_T locations, $\{x_c^i\}$, by sampling h_c. This gives us N_T integrals

$$T_{D,\{x_c^i\}}(\theta) \triangleq \frac{1}{h_c(x_c^i)} \int dG_c \; H(G_c) \frac{P(G_c \mid D)}{H(G_c)} F(G_c(x_c^i), q_\theta(x_c^i)). \tag{22}$$

Sometimes, these integrals also can be evaluated algebraically, giving closed form sample functions of θ. As an example, suppose we sample oracles according to the posterior, that is, take $H(G_c) = P(G_c \mid D)$, so that

$$T_{D,\{x_c^i\}}(\theta) = \frac{1}{h_c(x_c^i)} \int dG_c \; P(G_c \mid D) F(G_c(x_c^i), q_\theta(x_c^i)). \tag{23}$$

Next, say that we have a Gaussian process prior over oracles, $P(G_c)$ (Rasmussen and Williams, 2006), with a Gaussian covariance kernel. For this choice, and for some F's,

we can compute $T_{D,\{x_c^i\}}(\theta)$ exactly for any θ, provided D is not too large. For example, this is the case for most F's whose dependence on their first argument lies in the exponential family.[16] In other situations, while we cannot evaluate them exactly, the integrals $T_{D,\{x_c^i\}}(\theta)$ can be accurately approximated algebraically. This again reduces them to closed form sample functions of θ.

In either of these cases, there is no need to sample H; samples of h_c suffice. More generally, we may not be able to evaluate the N_T functions $T_{D,\{x_c^i\}}(\theta)$ algebraically, and also cannot form accurate low-order algebraic approximations to them. In this situation, for each x_c^i, we should generate one sample of G_c from $H(G_c)$.[17] This provides us a total of N_T sample functions

$$T_{D,\{x_c^i,G_c^i\}}(\theta) \triangleq \frac{P(G_c^i \mid D)F(G_c^i(x_c^i), q_\theta(x_c^i))}{h_c(x_c^i)H(G_c^i)}. \tag{24}$$

As an example, say we believe firmly that a particular posterior $P(G_c \mid D)$ governs our problem, and can sample that posterior. Then we can take $H(G_c)$ to equal $P(G_c \mid D)$. Now Eq. 24 only requires that we have values of our sampled G_c's at the $\{x_c^i\}$, that is, we only need to have values $G_c^i(x_c^i)$, where $G_c^i \sim H(G_c)$. So we only need to sample the N_T separate one-dimensional distributions $\{P(G_c(x_c^i) \mid D)\}$. In particular, we might be able to use Gaussian Process techniques to generate those values. Alternatively, as a rough approximation, one could simply fit a regression to the data in D, $\omega(x)$, and then add noise to the vector of values $\{\omega(x_c^i)\}$ to get $\{G_c^i(x_c^i)\}$. If we wanted to have multiple G_c's for each x_c^i, then we would simply generate more samples of each distribution $\{P(G_c(x_c^i) \mid D)\}$.[18]

However we sample H, the resultant sample functions provide the estimate

$$\begin{aligned} \hat{\tilde{\mathscr{F}}}_{D,\{x_c^i,G_c^i\}}(\theta) &\triangleq \frac{1}{N_T}\sum_{i=1}^{N_T} T_{D,\{x_c^i,G_c^i\}}(\theta) \\ &\approx \mathbb{E}(\tilde{\mathscr{F}}_{D,\{x_c^i,G_c^i\}}(\theta) \mid D) \end{aligned} \tag{25}$$

which we sometimes abbreviate as $\hat{\tilde{\mathscr{F}}}_D(\theta)$. For the situation where we can evaluate the integral over G_c algebraically (or at least approximate it that way), we instead define

$$\hat{\tilde{\mathscr{F}}}_D(\theta) \triangleq \frac{1}{N_T}\sum_{i=1}^{N_T} T_{D,\{x_c^i\}}(\theta) \tag{26}$$

and rely on the context to decide which of the definitions of $\hat{\tilde{\mathscr{F}}}_D(\theta)$ is meant. So for either case we can write $\tilde{\mathscr{F}}_D(\theta) \approx \hat{\tilde{\mathscr{F}}}_D(\theta)$.[19]

16. Strictly speaking, since the oracle is noise-free, the likelihood $P(D \mid G_c)$ is a delta function about having D lie exactly on the function $G_c(x)$. In practice, this may make the computation be ill-behaved numerically. Typically such problems are addressed modeling the fictitious oracles as though the values they returned had a small amount of Gaussian noise added.

17. One could have the number of samples of $H(G_c)$ not match the number of samples of $h_c(x)$. To avoid the associated notational overhead, here we just match up the two types of samples, one-to-one.

18. As an alternative, we could reverse the sampling order and sample $P(G_c \mid D)$ first and h_c second. Practically, this would mean generating N_T samples of h_c, and then sampling the single N_T-dimensional distribution $P(G_c(x_c^1), \ldots G_c(x_c^{N_T}) \mid D)$. (This contrasts to the case considered in the text in which H is sampled second, so one instead samples N_T separate one-dimensional distributions $\{P(G_c(x_c^i) \mid D)\}$.) If we do this using a 'rough approximation' based on a fit ω to D, the noise values added to the values of the fit, $\{\omega(x_c^i)\}$, would have to be correlated with each other, since they reflect the same G_c.

19. As a practical issue, we may want to divide the sum in Eq. 26 by the empirical average $\sum_{i=1}^{N_T} \frac{P(G_c^i|D)}{h_c(x_c^i)}$. Similarly, if we cannot evaluate the integral over G_c algebraically, we may want to modify the estimate

In both cases, under naive MCO, we search for the θ that minimizes $\hat{\tilde{\mathscr{F}}}_D(\theta)$. We then use that θ as our estimate for the solution to (P6). More generally, rather than use naive MCO we can exploit our sample functions with PLMCO. For example, rather than minimizing $\hat{\tilde{\mathscr{F}}}_D(\theta)$, we could minimize a sum of $\{\tilde{\mathscr{F}}_D(\theta)$ and a regularization penalty term.

However we arrive at our (estimated) optimal q_θ, most simplistically, we can update h to equal that new q_θ. In a more sophisticated approach, we could set h from the sample functions using active learning (see Sec. 6.6 below). Once we have that new h we can form samples of it to generate new factual sample locations x. These in turn are fed to the factual oracle G to augment our data set D. Then the process repeats.

Note that unlike with non-FB immediate sampling, with FB immediate sampling we need to evaluate $P(G_c \mid D)$ (or sample it, if we choose to have $H(G_c) \triangleq P(G_c \mid D)$). This may be non-trivial. On the other hand, that very same distribution $P(G_c \mid D)$ that may cause difficulty also gives the major advantage of the fit-based approach; it allows any insights we have into how to fit a curve G_c to the data points D to be exploited.

6.4 Exploiting FB Immediate Sampling

To illustrate how fitting might improve immediate sampling, consider the case where $\mathscr{F}_G(q_\theta)$ is qp KL distance. Say that $G(x)$ is a high-dimensional convex paraboloid inside a hypercube, and zero outside of that hypercube. Suppose as well that we have a single factual sampling distribution h, which is concentrated on one side of the paraboloid. For example, if the peak of the paraboloid is at the origin, h might be a Gaussian (masked by the hypercube) whose mean lies several sigmas away from the origin.

To start, consider importance sampling MC estimation of the integral $\mathscr{F}_G(q_\theta)$ for one particular θ, without any concern about choosing among θ's. Say that the factual sample D isn't too large. Then it is likely that no elements of D are in regions where G reaches its lowest values. For such a D, the associated factual estimate

$$\hat{\mathscr{F}}_D(\theta) \quad \triangleq \quad \frac{\sum_i r_h^i(x^i, \theta)}{N} \tag{27}$$

is larger than the actual value, $\mathscr{F}_G(q_\theta)$ (cf. Eq. 14). So straightforward importance-sampling integral estimation is likely to be badly off.[20]

Intuitively, the problem is that as far as the factual estimate $\hat{\mathscr{F}}_D(\theta)$ is concerned, G could just as well be a sum of delta functions centered at the x's in D, with low associated oracle sample values, as a paraboloid. If G were in fact such a sum, then $\hat{\mathscr{F}}_D$ would be correct. However by looking at the $(x, G(x))$ pairs in D, all of which lie on the same paraboloid, such an inference of G appears quite unreasonable. It makes sense to instead infer that G is a paraboloid.

Fitting is a way to formalize (and exploit) such D-based insights. As an example, consider using a Bayesian PL algorithm to do the fitting. Typical choices for the prior $P(G_c)$ used in PL would result in a posterior $P(G_c \mid D)$ that would be far more tightly concentrated about the actual G's paraboloid shape than about the sum of delta functions. Fitting would automatically reflect this, and thereby produce a better estimate of $\mathscr{F}_G(\theta)$ than $\hat{\mathscr{F}}_D(\theta)$.[21]

---

in Eq. 25 by dividing by $\sum_{i=1}^{N_T} \frac{P(G_c^i|D)}{H(G_c^i)h_c(x_c^i)}$. Such divisions would accord with the analogous division we do in our non-FB immediate sampling experiments.

20. Since importance sampling is unbiased, this means its variance is likely to be large.

21. It might be objected that in a different problem G actually would be the sum of delta functions, not the paraboloid. In that case the FB estimate is the one that would be in error. However this possibility is

Now we aren't directly concerned with the accuracy of our estimate $\mathscr{F}_G(q_\theta)$ for any single θ. We aren't even concerned with the overall accuracy of that estimate for a set of θ's. Rather we are concerned with the accuracy of the ranking of the θ's given by those estimates. For example, consider naive MCO, under which we choose the θ minimizing $\tilde{\mathscr{F}}_D(\theta)$. Even if all of our estimates (one for each θ) were far from the associated actual values, if their signed errors were identical, the naive MCO would perform perfectly.

In other words, ultimately we are interested in correlations between errors of our estimates of $\mathscr{F}_G(q_\theta)$ for different θ's. (See Sec. 2.3.) Nonetheless, we might expect that if we tend to have large error in our estimates of $\mathscr{F}_G(q_\theta)$ for the θ's, then everything else being equal, we would be likely to have large error in the associated estimate of an optimal θ.

In Sec. 4.5 we exploited the equivalence between PL and MCO to improve upon naive MCO. However the parameter in MCO doesn't specify a functional fit to a data set. Accordingly, the incorporation of PL into MCO considered in Sec. 4.5 doesn't involve fitting a function to D. This is why those PLMCO techniques don't address the issue raised in this example; fitting does that. So in full FB MCO, we may use PL in two separate parts of the algorithm, both to form the fit to D, and then to use those fits to choose among the θ's.

6.5 Statistical Analysis of FB MC

Before analyzing expected performance of FB MCO, we start with the simpler case of FB MC introduced at the beginning of this section. For simplicity we assume that the integral $\tilde{\mathscr{L}}_D$ can be calculated exactly for any D, so that no fictitious samples arise.

As discussed in Sec. 2.3, two important properties of an MC estimator of an integral $\mathscr{L}(\phi) = \int dw\ U(w, \phi)$ are the sample bias and the sample variance of that estimator. Together, these give the expected loss of the estimator under a quadratic loss function, conditioned on a fixed oracle $U(., \phi)$.

This is just as true for a Bayesian fitting algorithm as for any other. For quadratic loss, for sample set $D \equiv \{w^i, U(w^i, \phi)\}$, the Bayesian FB MC prediction for $\mathscr{L}$ is the posterior mean,

$$\tilde{\mathscr{L}}_D \quad = \quad \int dw'\ dU_c(., \phi)\ U_c(w', \phi) P(U_c(., \phi) \mid D). \tag{28}$$

Accordingly, the expected quadratic loss of Bayesian FBMC is

$$\int dD\ P(D \mid v, U(., \phi))[\mathscr{L} - \tilde{\mathscr{L}}_D]^2 \quad =$$

$$\int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i)[\int dw'\ U(w', \phi) \ - \ \int dw'\ dU_c(, \phi)\ U_c(w', \phi) P(U_c(., \phi) \mid D)]^2 \tag{29}$$

where v is the proposal distribution that is IID sampled N times to generate the sample set.

In the usual way one can re-express this expected quadratic loss using a bias-variance decomposition. Whereas a conventional importance sample estimator of $\int dw\ U(w, \phi)$ is unbiased, the Bayesian estimator is biased in general; typically

$$\int dD\ P(D \mid v, U(., \phi))\ \tilde{\mathscr{L}}_D \quad \neq \quad \mathscr{L}. \tag{30}$$

exactly what the prior $P(G_c)$ addresses; if in fact you have reason to believe that a G that is a sum of delta functions is *a priori* just as likely as a paraboloid G, then that should be reflected in $P(G_c)$. Doing so would in turn make the FB estimate more closely track the non-FB estimate.

This bias is a general characteristic of Bayesian estimators. Furthermore, for some functions $U(.,\phi)$, the Bayesian estimator will both be biased (unlike the factual sample estimator) and have higher variance than the factual sample estimator. So for those $U(.,\phi)$, the Bayesian estimator has worse bias plus variance.

In conventional importance sampling estimation of an integral, the sampling distribution v is used twice. First it is used to form the sample set. Then, when the sample set has been formed, v is used again, to set the denominator values in the ratios giving the MC estimate of the integral (cf. Sec. 2.2). In contrast, Bayesian FB MC doesn't care what v is. $P(U_c(.,\phi) \mid D)$ is independent of the values $v(w^i)$. As mentioned at the beginning of this section, this is a typical feature of FB MC estimators.

This feature does not mean that the sampling distribution is immaterial in FB MC however. Even though it does not arise in making the estimate, as Eq. 29 shows, v helps determine what the expected loss will be. Indeed, in principle at least, Eq. 29 can be used to guide the choice of the sampling distribution for Bayesian FB MC. It can even be used this way dynamically, at a midpoint of the sampling process, when one already has some samples of $U(.,\phi)$. Such a procedure for using Eq. 29 to set v dynamically amounts to what is called 'active learning' in the PL literature (see Freund et al., 1997; Dasgupta and Kalai, 2005).

We now generalize the foregoing to the case of a non-quadratic loss function L. The Bayesian estimator produces the estimate

$$\tilde{\mathscr{L}}_D \quad \triangleq \quad \text{argmin}_{\rho \in \mathbb{R}}[\int dU_c(.,\phi)\ P(U_c(.,\phi) \mid D) L[\tilde{\mathscr{L}}_D,\ \ \mathscr{L}_{U_c}]] \tag{31}$$

Given that the factual oracle is $U(.,\phi)$, the expected loss with that Bayesian estimator is

$$\int dw^1 \ldots dw^N \ \prod_{i=1}^{N} v(w^i) L[\tilde{\mathscr{L}}_{\{w^i, U(w^i,\phi)\}},\ \ \int dw'\ U(w',\phi)]. \tag{32}$$

The expected loss in Eq. 32 is an average over data with the oracle held fixed. This contrasts with the analogous quantity typically considered in Bayesian analysis, which is an average over oracles with the data held fixed. That quantity is the posterior expected loss,

$$\int dU(.,\phi) P(U(.,\phi) \mid D) L[\tilde{\mathscr{L}}_D, \mathscr{L}_U(\phi)] \tag{33}$$

In general, different $U(.,\phi)$'s will give different risks for the same estimator. So we can adapt any measure concerning loss in which $U(.,\phi)$ varies, to concern risk instead. In particular, the posterior expected risk is

$$\int dU(.,\phi) P(U(.,\phi) \mid D)\ \{L[\tilde{\mathscr{L}}_D, \mathscr{L}_U(\phi)]\ -\ \text{min}_{\rho \in \mathbb{R}}[L[\rho, \mathscr{L}_U(\phi)]]\}. \tag{34}$$

Often the lower bound on loss is always 0, so that $\text{min}_{\rho \in \mathbb{R}}[L[\rho, \mathscr{L}_U(\phi)]] = 0\ \forall\ U(.,\phi)$. In this case posterior expected risk just equals posterior expected loss.

We can combine the non-Bayesian and Bayesian analyses, involving expected loss and posterior expected loss respectively. To do this we consider the prior-averaged expected loss, given by

$$\int dU(.,\phi)\ P(U(.,\phi)) \int dw^1 \ldots dw^N \ \prod_{i=1}^{N} v(w^i) L[\tilde{\mathscr{L}}_{\{(w^i, U(w^i,\phi))\}},\ \int dw'\ U(w',\phi)]. \tag{35}$$

where $P(U(.,\phi))$ is a prior distribution over oracles.

Note that the prior-averaged expected loss is an average over both oracles and sample sets. It reflects the following experimental test of our FB MCO algorithm: Multiple times a factual oracle $U(.,\phi)$ is generated by sampling $P(U(.,\phi))$. For each such $U(.,\phi)$, many times a factual sample set D is generated by sampling the likelihood $P(D \mid U(.,\phi), v)$. That D is then used by the FMCO algorithm to calculate $\mathscr{L}_D$. In performing that calculation, the algorithm assumes the same likelihood as was used to generate D, but its prior $P(U_c(.,\phi))$ may not be the same function of $U_c(.,\phi)$ as $P(U(.,\phi))$ is of $U(.,\phi)$. Then the loss between $\mathscr{L}_D$ and $\mathscr{L}_U$ is calculated. The quantity in Eq. 35 is the average of that loss.

Say that $P(U(.,\phi))$ is the same function of $U(.,\phi)$ as $P(U_c(.,\phi))$ is of $U_c(.,\phi)$. Then the Bayesian estimator is based on the actual prior. In this case, the Bayesian estimator $\tilde{\mathscr{L}}_D$ will minimize the prior-averaged expected loss of Eq. 35.[22] In general though, there is no reason to suppose that these two priors are the same. In the real world where those priors differ, expected loss for a Bayesian estimator is given by an inner product between the posterior used by that estimator, $P(U_c(.,\phi) \mid D)$, and the true posterior, $P(U(.,\phi) \mid D)$ (see Wolpert, 1997, 1996).[23]

As before, since $U(.\phi)$ varies in the integrand of prior-averaged expected loss, we can can adapt it to get a prior-averaged expected risk. This is given by

$$\int dU(.,\phi)\ P(U(.,\phi)) \int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i) \quad \times$$
$$\{L[\tilde{\mathscr{L}}_{\{(w^i, U(w^i,\phi))\}},\ \mathscr{L}(\phi)] \quad - \quad \min_{\rho \in \mathbb{R}}[L[\rho,\ \mathscr{L}(\phi)]]\}. \tag{36}$$

As before, if the minimal loss is always 0, then prior-averaged expected risk just equals prior-averaged expected loss.

Broadly speaking, in Bayesian approaches to Monte Carlo problems, the sampling distribution that generated the samples is immaterial once one those samples have been generated (see Rasmussen and Gharamani, 2003) and references therein). So what difference does the choice of a sampling distribution like v make to a Bayesian? The answer is that v determines how likely it is that we will generate a D with a high posterior variance of the quantity of interest. For example, say one wishes to form an importance sampling estimate of $\mathscr{L} = \int dx\, U(x)$ using sampling distribution v to generate sample set D. Then if one changes v, one changes the likelihoods of the possible D. Moreover, each D has its own posterior variance, $\mathrm{Var}(\mathscr{L}_c \mid D)$. So what a good choice of v means is that a D with poor $\mathrm{Var}(\mathscr{L}_c \mid D)$ is unlikely to be formed, that is, that $\int dD\ P(D \mid v)\mathrm{Var}(\mathscr{L}_c \mid D)$ is low.

---

22. To see this, replace $\tilde{\mathscr{L}}_D$ with some arbitrary function of D, $f(D)$. Our task it to solve for the optimal f. First interchange the integrals over data and over oracles in Eq. 35. Next consider the integrand of the outer (data) integral,

$$\int dU(.,\phi)\ P(U(.,\phi)) \prod_{i=1}^{N} v(w^i) L[f(D),\ \int dw'\ U(w',\phi)]].$$

Since we are considering a noise-free oracle, we can write this as

$$\int dU(.,\phi)\ P(U(.,\phi)) P(D \mid v, U(.,\phi)) L[f(D), \mathscr{L}_U(\phi)].$$

Since $P(U(.,\phi))P(D \mid v, U(.,\phi)) \propto P(U(.,\phi) \mid v, D) = P(U(.,\phi) \mid D)$, this integral is minimized by setting $f(D) = \tilde{\mathscr{L}}_D$. **QED.**

23. It is in recognition of the fact that those functions might differ that we have been referring to 'Bayesian' rather than 'Bayes-optimal' estimators.

6.6 Statistical Analysis of FB MCO

We can extend the statistical analysis of FB MC to the case of FB MCO by allowing ϕ to vary. The Bayesian choice of ϕ is the one that minimizes posterior expected loss,

$$\tilde{\phi}_D \triangleq \text{argmin}_\phi[\int dU_c \; P(U_c \mid D) L(\phi, U_c)]. \tag{37}$$

Since $P(U_c \mid v, D) = P(U_c \mid D)$, this estimator is independent of v. The same is true for the posterior expected loss of this Bayesian estimator,

$$\int dU \; P(U \mid D) L(\tilde{\phi}_D, U). \tag{38}$$

On the other hand, the expected loss associated with this estimator,

$$\int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i) L(\tilde{\phi}_{\{w^i, U(w^i)\}}, U), \tag{39}$$

explicitly depends on v. So does the prior-averaged expected loss,

$$\int dU \; P(U) \int dw^1 \ldots dw^N \prod_{i=1}^{N} v(w^i) L(\tilde{\phi}_{\{w^i, U(w^i)\}}, U). \tag{40}$$

Next, the posterior expected risk is

$$\int dU \; P(U \mid D) \{L(\tilde{\phi}_D, U) \; - \; \min_{\phi'}[L(\phi', U)]\} \tag{41}$$

where ϕ' runs over the (implicit) set of all possible ϕ. In general $\min_{\phi'}[L(\phi', U)]$ varies with U. (For example, this is the case with the loss function $\mathscr{L}_U(\phi)$ of Eq. 4.) Accordingly, unlike in Bayesian FB MC, typically in Bayesian FB MCO the posterior expected risk does not equal the posterior expected loss.

Finally, the prior-averaged expected risk is

$$\int dU dw^1 \ldots dw^N \; P(U) \prod_{i=1}^{N} v(w^i) \{L(\tilde{\phi}_{\{w^i, U(w^i)\}}, U) \; - \; \min_{\phi'}[L(\phi', U)]\}. \tag{42}$$

Again, since $\min_{\phi'}[L(\phi', U)]$ typically varies with U, in general this prior-averaged expected risk does not equal the prior-averaged expected loss. However the estimator that minimizes prior-averaged expected loss — $\tilde{\phi}_D$ — is the same as the estimator that minimizes prior-averaged expected risk.[24]

For any particular fitting algorithm, our equations tell us how performance of the associated FB MCO depends on v and either $P(U(., \phi))$ or the pair $P(U(., \phi))$ and $P(U_c(., \phi))$, depending on which equation we consider. So if we fix those prior(s), our equations tell us, formally, what the optimal v is.

One can consider estimating that optimal v at a mid-way point of the algorithm, based on the algorithm's behavior up to that point. One can then set v to that estimate for the remainder of the algorithm.[25] Doing this essentially amounts to a type of active learning.

24. This follows from the fact that the prior-averaged lowest possible risk, the term subtracted in Eq. 42, is independent of the choice of the estimator.

25. Note though that if one intends to update v more than once, then strictly speaking the first update to v should take into account the fact that the future update will occur. That means the equations above for expected loss, prior-averaged expected loss, etc., no longer apply.

As with Bayesian FBMC, we can analyze the effects of having $P(U)$ not be the same function of U as $P(U_c)$ is of U_c. Since PL and MCO are formally the same, such an analysis applies to parametric machine learning in addition to FB MCO. In particular, the analysis gives a Bayesian correction to the bias-variance decomposition of supervised learning. This correction holds even if the fitting algorithm in the supervised learning cannot be cast as Bayes-optimal for some assumed prior $P(U_c)$. Intuitively speaking, the correction means that the bias-variance decomposition gets replaced by a bias-variance-covariance decomposition. That covariance is between the posterior distribution over target functions on the one hand, and the posterior distribution over fits produced by the fitting algorithm on the other (see Wolpert, 1997).

6.7 Combining FB and Non-FB Estimates in FB MCO

Return now to the example in Sec. 6.6, where the factual sample is formed by importance sampling the factual oracle and we form a fictitious sample set using fictitious oracles.Then using only D, our estimate of $\mathscr{F}_G(\theta)$ would be the factual estimate, $\hat{\mathscr{F}}_D(\theta) = \frac{\sum_{k=1}^{N} r^k(\theta)}{N}$. Using only our fictitious samples would instead give us the estimate $\hat{\tilde{\mathscr{F}}}_D(\theta)$.

On the one extreme, say we firmly believe that distribution we use for the posterior $P(G_c \mid D)$ is correct. (So in particular we firmly believe that the factual oracle G was generated by sampling the prior $P(G_c)$.) Then in the limit $N_T \to \infty$, $\forall\theta$ our importance-sample estimate of $\tilde{\mathscr{F}}_D$ will be exactly correct. So Bayesian decision theory would direct us to use the associated estimate $\hat{\tilde{\mathscr{F}}}_D(\theta)$, and ignore $\hat{\mathscr{F}}_D(\theta)$. At the other extreme, say that $N_T = 1$, while N, the number of factual samples, is quite large. In such a situation, even if we believe our posterior is correct, it would clearly be wrong to use $\hat{\tilde{\mathscr{F}}}_D(\theta)$ as our estimate, ignoring $\hat{\mathscr{F}}_D(\theta)$.

How should we combine the estimates in this latter situation? More generally, even when we believe our posterior is correct, unless the number of fictitious samples is far greater than the number of factual samples, we should combine the two associated estimates. How best to do that? Does the fact that $\hat{\tilde{\mathscr{F}}}_D(\theta)$ is estimated via importance sampling over a much larger space than $\hat{\mathscr{F}}_D(\theta)$ affect how we should combine them? More generally, say we don't presume that our $P(G_c \mid D)$ is exactly correct; how should we combine the estimates then?

One is tempted to invoke Bayesian reasoning to determine how best to combine the two estimates. While that might be possible in certain situations, often determining the optimal Bayesian combination would necessitate yet more Monte Carlo sampling of some new integrals. It would be nice if some other approach could be used.

One potential such approach is stacking (Wolpert, 1992; Breiman, 1996; Smyth and Wolpert, 1999). In this approach, one many times partitions the factual sample D into two parts, a 'training set' D^1, and a 'validation set' D^2. We write the values of w and U in D^1 as $\{D^1_w(i)\}$ and $\{D^1_U(i,\phi)\}$ respectively, and similarly for D^2. For each such partition one would run both the non-FB MCO algorithm and the FB MCO algorithm on D^1. That generates the estimates $\hat{\phi}_{v,U,D^1}$ and $\hat{\tilde{\phi}}_{D^1}$, respectively.

Those two ϕ's give us two associated error values on the validation set, $\sum_j D^2_U(j, \hat{\phi}_{v,U,D^1})$ and $\sum_j D^2_U(j, \hat{\tilde{\phi}}_{D^1})$, respectively. More generally, we can evaluate the error on the the validation set of *any* ϕ, in addition to the errors of $\hat{\phi}_{v,U,D^1}$ and $\hat{\tilde{\phi}}_{D^1}$. Moreover, we can do this for the validation set of any of the partitions of D. Note, however, that *only* factual samples are used for cross-validation.

This is what stacking exploits. In the most straightforward use of stacking, one searches for a function mapping the ϕ's produced by our two algorithms to a composite ϕ. The goal is to find such a composite ϕ that will have as small validation set error (when averaged over all partitions) as possible.

For example, if ϕ is a Euclidean vector, one could perform a regularized search for the weighted sum of ϕ's that gives minimal partition-averaged validation set error. Let the weights produced by that search be b_{FB} and b_{non-FB}. Then to find the final estimate for ϕ, one would use those weights to sum the outputs of the algorithms when run on all of D: $b_{non-FB}\hat{\phi}_{v,U,D} + b_{FB}\tilde{\phi}_D$.

7. Conclusion

In this paper we explored the relationship between Monte Carlo Optimization of a parametrized integral, parametric machine learning, and 'blackbox' or 'oracle'-based optimization. We made four contributions.

First, we proved that MCO is identical to a broad class of parametric machine learning problems. This should open a new application domain for previously investigated parametric machine learning techniques, to the problem of MCO.

To test the use of PL in MCO one needs an MCO problem domain. The one we used was based on our second contribution, which was the introduction of immediate sampling. Immediate sampling is a way to transform an arbitrary blackbox optimization problem into an MCO problem. Accordingly, it provides us a way to test the use of PL to improve MCO, but testing whether it can improve blackbox optimization.

In our third contribution we validated this way of improving blackbox optimization. In particular, we demonstratied that cross-validation and bagging improve immediate sampling.

Conventional Monte Carlo and MCO procedures ignore some features of the sample data. In particular, they ignore the relationship between the sample point locations and the associated values of the integrand; only the values of the integrand at those locations are considered. We ended by presenting fit-based MCO, which is a way to exploit the information in the sample locations.

There are many PL techniques that should be applicable to immediate sampling but that are not experimentally tested in this paper. These include density estimation active learning, stacking, kernel-based methods, boosting, etc. Current and future work involves experimental tests of the ability of such techniques to improve MCO in general and immediate sampling in particular.

Other future work is to conduct experimental investigations of the three techniques that we presented in this paper but did not test. One of these is fit-based MCO (and fit-based immediate sampling in particular). The other two are the techniques described in the appendices: immediate sampling for constrained optimization problems, and immediate sampling with elite objective functions.

There are also many potential application domains for immediate sampling PC for blackbox optimization that we intend to explore. Some of these exploit the ability of such PC to handle arbitrary (mixed) data types of x's. In particular, one such data type is the full trajectory of a system through a space; for optimizing a problem over such a space, PC becomes a form of reinforcement learning.

A. Constrained Optimization

Under the PC transform we replace an optimization problem over X with one over $\mathscr{Q}$. As discussed at the beginning of Sec. 3.3, the characteristics of the transformed objective can be very different from those of the original objective.

Similarly, characteristics of any constraints on X in the original problem can also change significantly under this transformation. More precisely, say we add to (P4) equality and inequality constraints restricting $x \in X$ to a **feasible region**. Then to satisfy those X-constraints we need to modify (P5) to ensure that the support of the solution $q_\theta(.)$ is a subset of the feasible region in X.

This appendix considers some ways of modifying PC to do this. For earlier work on this topic in the context of delayed sampling, see Wolpert et al. (2006); Bieniawski and Wolpert (2004); Bieniawski et al. (2004); Macready and Wolpert (2005).

A.1 Guaranteeing Constraints

Say we have a set of equality and inequality constraints over X. Indicate the feasible region by a feasibility indicator function

$$\Phi(x) = \begin{cases} 1, & x \text{ is feasible,} \\ 0, & \text{otherwise.} \end{cases}$$

For simplicity, we assume that for any x, we can evaluate $\Phi(x)$ essentially 'for free'.

The transformed version of this constrained optimization problem is

$$(\text{P5}_c): \quad \begin{array}{ll} \text{minimize} & \mathscr{F}_{\mathscr{G}}(q_\theta), \\ \text{subject to} & q_\theta(x)\Phi(x) = 0. \end{array}$$

We now present a parametrization for q that ensures that it has zero support over infeasible x. First, let $\tilde{q}$ be any parametrized distribution over X, for instance, a mixture of Gaussians. Then using $\Phi(x \in X)$ as a 'masking funtion' we parametrize $q_\theta(x)$ as

$$\begin{aligned} q_\theta(x) &\triangleq \frac{\tilde{q}_\theta(x)\Phi(x)}{\int dx' \; \tilde{q}_\theta(x')\Phi(x')} \\ &\triangleq \tilde{q}_{\Phi,\theta}(x). \end{aligned}$$

This q_θ automatically meets the constraints; it places zero probability mass at infeasible x's. It transforms the constrained problem $(\text{P5})_c$ into the unconstrained problem

$$(\text{P5}_{uc}) : \text{minimize } \mathscr{F}_{\mathscr{G}}(\tilde{q}_{\Phi,\theta}).$$

Now consider the case where $\mathscr{F}_{\mathscr{G}}$ is an integral over X. Typically in this case we are only concerned with the values of the associated integrand at feasible x's. For example, when $E_{q_\theta}(G)$ is of interest, it's usually because our ultimate goal is to find a feasible x with as good a $G(x)$ as possible. In this situation it makes no sense to choose between two candidate q_θ's based on differences in (the G values at) the regions of infeasible x that they emphasize. More formally, our choice between them should be independent of their respective values of $\int dx \; [1 - \Phi(x)]q_\theta(x)G(x)$. We can enforce this by replacing the objective $E_{q_\theta}(G) = \int dx \; q_\theta(x)G(x)$ with $\int dx \; \Phi(x)q_\theta(x)G(x)$. If we then use the barrier function approach outlined above, our final objective becomes qp KL distance with the integral restricted to feasible x's.

Generalizing this, when we are not interested in behavior at infeasible x we can reduce the optimization problem further from (P5_{uc}), by restricting the integral to only run over

feasible x's. More precisely, write the original problem (P5$_c$) as the minimization of

$$\int dx[\int dg P(g \mid x, \mathscr{G})]F(g, q_\theta(x)) \quad \triangleq \quad \int dx\ \mu(x, q_\theta(x)),$$

subject to the constraints on the support of q_θ. By using the $\tilde{q}$ construction we can replace this constrained optimization problem with the unconstrained problem

$$\begin{aligned} \text{(A1): } \text{argmin}_q \int dx\ \Phi(x)\mu[x, q_\theta(x)] &= \text{argmin}_\theta \int dx\ \Phi(x)\mu[x, \tilde{q}_{\Phi,\theta}(x)], \\ &= \text{argmin}_\theta \int dx\ \Phi(x)\mu[x, \frac{\Phi(x)\tilde{q}(x)}{\int dx'\Phi(x')\tilde{q}(x')}]. \end{aligned}$$

As an example, say our original objective function is pq KL distance. Define $Z_\Phi^\beta \equiv \int dx\ p^\beta(x)\Phi(x)$. Then our new optimization problem is to minimize over θ

$$\begin{aligned} \text{KL}(p_\Phi^\beta \mid\mid \tilde{q}_{\Phi,\theta}) &= \text{KL}(\frac{p^\beta \Phi}{Z_\Phi^\beta} \mid\mid \tilde{q}_\Phi) \\ &= -\int dx\ \frac{\Phi(x)p^\beta(x)}{Z_\Phi^\beta} \ln[\frac{\tilde{q}_\theta(x)\Phi(x)}{\int dx'\tilde{q}_\theta(x')\Phi(x')}], \\ &= -\int dx\ \frac{\Phi(x)p^\beta(x)}{Z_\Phi^\beta}\ \{\ln[\tilde{q}_\theta(x)] + \ln[\Phi(x)] - \ln[\int dx'\tilde{q}_\theta(x')\Phi(x')]\}. \end{aligned}$$

The $\tilde{q}_\theta$ minimizing this is the same as the one that maximizes

$$\int dx \frac{\Phi(x)p^\beta(x)}{Z_\Phi^\beta} \ln[\tilde{q}_\theta(x)] \quad - \quad \ln[\int dx'\ \tilde{q}_\theta(x')\Phi(x')]. \tag{43}$$

We can estimate Z_Φ^β using MC techniques. We can then apply MCO to estimate the θ that maximizes the integral difference[26] in Eq. 43.

To generate a sample of sample $q_\theta(x) = \tilde{q}_\theta(x)\Phi(x)$ we can subsample[27] $\tilde{q}_\theta$ according to Φ. In some cases though, this can be very inefficient (that is, one may get many rejections before getting a feasible x). To deal with such cases, we can first run a density estimator on the samples of feasible x's we have so far, getting a distribution π. (Note that no extra calls to the feasibility oracle are needed to do this.) Next write $q_\theta(x) = \pi(x)[q_\theta(x)/\pi(x)]$. This identity justifies the generation of samples of q_θ by first sampling $\pi(x)$ and then subsampling according to $q_\theta(x)/\pi(x) = \tilde{q}_\theta(x)\Phi(x)/\pi(x)$.

In an obvious modification to the foregoing, we can replace the hard restriction that supp(q) contain only feasible x's, with a 'soft' constraint that $q(x) \leq \kappa\ \forall$ infeasible x. A similar alternative is to 'soften' $\Phi(x)$ by replacing it with κ for all infeasible x, for some $\kappa > 0$. For either alternative we anneal κ down to 0, as usual, perhaps using cross-validation.

26. Note that in general this difference of integrals will not be convex in $\tilde{q}_\theta$ for product distributions, unlike $\text{KL}(p_\Phi^\beta \mid\mid \tilde{q}_\theta)$. See the discussion at the end of Sec. 3.1 on product distributions and pq distance.

27. Say we want to sample a distribution $A(x) \propto B(x)C(x)$ where B is a distribution and C is non-negative definite, with c some upper bound on C. To generate such a sample by 'subsampling B according to C' we first generate a random sample of $B(.)$, getting x'. We then toss a coin with bias $C(x')/c$. If that coin comes up heads, we keep x' as our sample of A. Otherwise we repeat the process (see Wolpert et al., 2006; Robert and Casella, 2004).

A.2 Alternative $\mathscr{F}_{\mathscr{G}}$

Since we're maximizing our expression over $\tilde{q}_\theta$, the second, correcting integral in Eq. 43 will tend to push $\tilde{q}_\theta$ to have probability mass *away* from feasible regions. To understand this intuitively, say that $\tilde{q}_\theta$ is a Gaussian and that the feasible region is 'spiky', resembling a multi-dimensional star-fish with a large central region and long, thin legs. For this situation, if we over-concentrate on keeping most of $\tilde{q}_\theta$'s mass restricted to feasible x, our Gaussian will be pushed away from any of the spikes of the feasible x's, and concentrate on the center. If the solution to our original optimization problem is in one of those spikes, such over-concentration is a fatal flaw. The second integral in Eq. 43 corrects for this potential problem.

More broadly, consider typical case behavior when one applies some particular constrained optimization algorithm to any of the problems in a particular class of optimization problems. As a practical matter, there is a spectrum of such problem classes, indexed by how difficult it is just to find feasible solutions on typical problems of the class. On the one side of this spectrum are problem classes where it is exceedingly difficult to find such a solution, e.g., high-dimensional satisfiability problems with a performance measure G superimposed to compare potential solutions. On the other end are "simple" problem classes where it is reasonable to expect to find a feasible solution. The 'starfish' optimization problem is an example of a problem of the former type.[28]

For problems on the first side of the spectrum, where just getting a substantial amount of probability mass into the feasible region is very difficult, we may want to leave out the second integral in Eq. 43. In other words, we may want to minimize $\mathrm{KL}(p_\Phi^\beta \mid\mid \tilde{q}_\theta)$ rather than $\mathrm{KL}(p_\Phi^\beta \mid\mid \tilde{q}_{\Phi,\theta})$. The reason to make this change is so that $\tilde{q}_\theta$ won't get pushed away from the feasible region. (As an aside, another potential benefit of this change is that if we make it, then for product distribution $\tilde{q}_\theta$, $\mathscr{F}_{\mathscr{G}}(.)$ is convex.)

Even if we do make this change, when we sample the resultant $\tilde{q}_\theta$ we may not get a feasible x. If this happens, a natural approach is to repeatedly sample $\tilde{q}_\theta$ until we do get a feasible x. However the resultant distribution of x's is the same as that formed by sampling $\tilde{q}_{\Phi,\theta}$ for the same θ. So under this 'natural approach' we work to optimize a distribution $(\tilde{q}_\theta)$ different from the one we ultimately sample $(\tilde{q}_{\Phi,\theta})$. This means that this approach may not properly balance our two conflicting needs for $\tilde{q}_\theta$: that it have most of its support in the feasible region, and that it be peaked about x's with high $p^\beta(x)$.

To illustrate this issue differently, take $\tilde{q}_\theta$ to be normalized, and to avoid multiplying and dividing by zero, modify $\Phi(x)$ to equal some very small non-zero value κ for infeasible x (as discussed above). Then under this 'compound procedure', we ultimately sample $\tilde{q}_{\Phi,\theta}(.)$. However we do not choose $\mathscr{F}_{\mathscr{G}}(\tilde{q}_{\Phi,\theta}) = \mathrm{KL}(p_\Phi^\beta \mid\mid \tilde{q}_{\Phi,\theta})$ as the function of θ that we want to minimize. Instead we choose

$$\mathscr{F}_{\mathscr{G}}(\tilde{q}_{\Phi,\theta}) \quad = \quad \mathrm{KL}(p_\Phi^\beta \mid\mid \tilde{q}_{\Phi,\theta}) \; - \; \ln[\int dx' \; \frac{\tilde{q}_{\Phi,\theta}(x')}{\Phi(x')}]. \tag{44}$$

A.3 Using Constraints for Unconstrained Optimization

Return now to unconstrained optimization problems. Say that we have reason to expect that over a particular region R, the distribution $p^\beta(x)$ has values approximately κ times as small as its value over $X \setminus R$. It would be nice to reflect this insight in our parametrization of q, that is, to parametrize q in a way that makes it easy to match it to $p^\beta(x)$ accurately. We can do this using a binary-valued function Φ and the approaches presented above.

28. Note that since we are discussing typical-case behavior, computational complexity considerations do not apply.

To illustrate this, define $\tilde{q}_{\Phi,\theta}$ as above and choose the objective function $\mathscr{F}_{\mathscr{G}}(\theta) = \mathrm{KL}(p^\beta \mid\mid \tilde{q}_{\Phi,\theta})$, where $\Phi(x) = \kappa$ over R, and equals 1 over $X \setminus R$.[29] Then working through the algebra, the q_θ that minimizes this objective is given by the $\tilde{q}_\theta$ that minimizes

$$\begin{aligned} -\int dx p^\beta(x) \ln[\tilde{q}_\theta(x)] \; + \; \ln[\int dx' \tilde{q}_\theta(x')\Phi(x')] &= \mathrm{KL}(p^\beta \mid\mid \tilde{q}_\theta) \; + \; \ln[\int dx' \tilde{q}_\theta(x')\Phi(x')] \\ &= \mathrm{KL}(p^\beta \mid\mid \tilde{q}_\theta) \; + \\ &\qquad \ln[\int dx'_R \; \kappa \tilde{q}_\theta(x') + \int dx'_{X\setminus R} \; \tilde{q}_\theta(x')]. \end{aligned} \tag{45}$$

The logarithm on the right-hand side is a 'correction' to pq distance from p^β to $\tilde{q}_\theta$, a correction that pushes $\tilde{q}_\theta$ *away* from regions where $\Phi(x) = 1$ (assuming $\kappa < 1$). To use immediate sampling with this parametrization scheme, once we find the $\tilde{q}_\theta$ that minimizes the sum of pq distance plus that correction term, we would set h to the distribution (proportional to) $\tilde{q}_\theta(x)\Phi(x)$. So we would generate our new samples from $\tilde{q}_\theta(x)\Phi(x)$, for example by subsampling.[30]

B. The Elite Objective Function

Not all PC objectives can be cast as an integral transform. Properly speaking, the choice of objective should be set by how the final q_θ will be used. For instance, the concept of expected improvement suggested by Mockus et al. (1978), and used by Jones et al. (1998), considers an objective (to be maximized) given by $\max(G_{\mathrm{bc}} - G(x), 0)$, where G_{bc} is the best of all the current samples, $\min_i\{G(x^i)\}$. This means that at each step we will take a single sample, and want to maximize the improvement. This is a simplification; even though the next sample may yield any improvement, it may be informative, so that we get a good sample ten steps later. A less simplistic objective is the following:

In blackbox optimization, no matter how many calls to the (factual) oracle we make, we will ultimately choose the best x (as far as the associated G value is concerned) out of all the ones that were fed to the oracle during the course of the entire run. Our true goal in BO is to have the G associated with *that* best x be as small as possible. For a discussion of distributions of extremal values, see Leadbetter et al. (1983); Resnick (1987).

Given that q_θ varies over the run in a way that we do not know beforehand, how can one approximate this goal as minimizing an objective function that is well-defined at all points during the run? One way to do this is to assume that there is some integer N such that, simultaneously,

1. It is likely that the best x will be one of the final N calls to the oracle during the run;
2. It is likely that q_θ will not vary much during the generation of those final N samples.

Under (2) we can approximate the q_θ's that are used to generate the final N calls as all being equal to some canonical q_θ. Under (1), our goal then becomes finding the canonical

29. Note that we use p^β in this $\mathscr{F}_{\mathscr{G}}$, not p^β_Φ, which is what we used for constrained optimization. This is because our goal now is simply to find a q_θ that matches $p^\beta(x)$. There are no additional aspects to the problem involving feasibility regions that have no *a priori* relation to $G(x)$.
30. In practice, $\Phi(x)$ for this unconstrained case would not be provided by an oracle. Instead we would typically have to estimate it. We could do that for example by using a regression to form a fit to samples of $p^\beta(x)$ and then use that regression to define the region R.

q_θ that, when sampled N times, produces a set of x's whose best element is as good as possible. [31]

In this appendix we make some cursory comments about this objective function, which we call the **elite objective function**. We focus on the use of Bayesian FB techniques with this objective. For a noise-free oracle the CDF for the elite objective is

$$\text{CDF}(k) \triangleq 1 - \int dx^1 \ldots dx^N \prod_{i=1}^{N} [q_\theta(x^i)\Theta(G(x^i) - k)]. \tag{46}$$

So the associated density function is

$$\begin{aligned} f(k) &= \frac{d\,\text{CDF}(k)}{dk} \\ &= N q_\theta(k)[\int dx\; q_\theta(x)\Theta(G(x) - k)]^{(N-1)}. \end{aligned} \tag{47}$$

The associated expectation value, $\int dk\; k f(k)$, is not linear in q_θ.

Writing it out, the posterior expected best-of-K value returned by the oracle when queries are generated by sampling q_θ is

$$\int dx^1 \ldots dx^K \prod_{i=1}^{K} q_\theta(x^i) \int dG\; P(G \mid D) \int dg^1 \ldots dg^K \prod_{j=1}^{K} P(g^k \mid x^k, G) \min_k\{g^k\} \tag{48}$$

We want the θ minimizing this. Say we knew the exact posterior $P(G \mid D)$ and could evaluate the associated integral in Eq. 48 closed-form. In this case there would be need for the parametric machine learning techniques used in the text. In particular there would be no need for regularization — an analogous role is played by the prior $P(G)$ underlying $P(G \mid D)$.

When we cannot evaluate the integral in closed form we must approximate it. To illustrate this, as in Sec. 6, for simplicity consider a single-valued oracle G. This reduces Eq. 48 to

$$\int dx^1 \ldots dx^K \prod_{i=1}^{K} q_\theta(x^i) \int dG\; P(G \mid D) \min_k\{G(x^k)\}. \tag{49}$$

(The analogous FB MCO equation for objective functions involving a single integral Eq. 20; here the single 'x' in that equation is replaced with a set of K x's sampled from q_θ.) To approximate this integral we draw N_T sample-vectors of K x's each, using a sampling distribution $h_c(x)$ to do so. At the same time we draw N_T fictitious oracles from some sampling distribution H over oracles.

31. An obvious variant of this reasoning is to have N vary across the run of the entire algorithm, at any iteration t being only the number of *remaining* calls to the oracle that we presume will be made. In this variant, one would modify the elite objective function to only involve the $N(t)$ remaining samples whose G value is better than the best found by iteration t. For the case $N = 1$, this is analogous to the expected improvement idea in Jones et al. (1998). Note that this variant objective function will change during the run, which may cause stability problems.

To simplify notation, let $\vec{x}$ indicate such a K-tuple of x's. (So for multidimensional X, $\vec{x}$ is actually a matrix.) Also write

$$\begin{aligned} h_c(\vec{x}) &\triangleq \prod_{i=1}^{K} h_c(x^k), \\ q_\theta(\vec{x}) &\triangleq \prod_{i=1}^{K} q_\theta(x^k), \\ G(\vec{x}) &\triangleq (G(x^1), \ldots, G(x^K)). \end{aligned} \tag{50}$$

With this notation, the estimate based on fictitious samples introduced in Sec. 6 becomes[32]

$$\sum_{i=1}^{N} \prod_{j=i}^{K} \frac{q_\theta(x_i^j)}{h_c(x_i^j)} \frac{P(G_i \mid D)}{H(G_i)} \min_{k=1,\ldots K}\{G_i(x_i^k)\}. \tag{51}$$

As discussed in Sec. 6, it is often good to set $H(G)$ to be as close to $P(G \mid D)$ as possible. So for example if we assume a Gaussian process model, typically we can set $H(G_i) = P(G_i \mid D)$, and then directly sample H to get the values of one G_i at the K separate points x_i^j. Alternatively, we can first form a fit $\phi(x)$ to the data in D. Next, for each of N samples $\vec{x}_i$, sample a colored (correlated) noise process over the K points $\{\vec{x}_i\}$ to get K real numbers. Finally, add those K numbers to the corresponding values $\{\phi(\vec{x}_i^j) : j = 1, \ldots, K\}$. This gives our desired sample of $\{G_i(\vec{x}_i)\}$.

To illustrate the foregoing, suppose $K = 1$, and that we have no regularization on q_θ. Then, in general, the sum in Eq. 51 is minimized by a q_θ that is a delta function about that data point x_i^1 with the best associated value $G_i(x_i^1)/h_c(x_i^1)$. However for $K > 1$, even without regularization, the optimal q_θ is *not* a delta function, in general.[33] In addition to the regularization-based argument in the text, this gives a more formal reason why the optimal q_θ should not be infinitely peaked.

When $K > 1$, the peakedness of q_θ parallels the peakedness of another non-negative function over x's, namely $P(G : G(x)$ is minimized at $x \mid D)$. However, if we run a few iterations of FB MCO with the elite objective, then D grows, and so $P(G : G(x)$ is minimized at $x \mid D)$ gets increasingly peaked over x's. (Intuitively, the larger D is, the more confident we are about G, and consequently the more confident we are about what regions of x's minimize G.) Accordingly, q_θ gets increasingly peaked as the algorithm progresses.

Note that this happens even though there is no external annealing schedule. This reflects the fact that the elite objective has no hyperparameter or regularization parameter like the β that appears in both the pq and qp objective functions.

C. Gaussian Example for Risk Analysis

The following example illustrates the foregoing for the case of Gaussian π, where only moments of π up to order 2 matter.

To illustrate the foregoing, consider the simple case where there are only two ϕ's, ϕ^1 and ϕ^2. Suppose that U and X are such that π is a two-dimensional Gaussian. Write π's mean as μ. Say that one of π's principal axes is parallel to the diagonal line, $l_1 = l_2$ (that

32. In practice there might be more efficient sampling procedures than Eq. 51. For example, one could form NK samples of $h_c(x)$ and N samples of $H(G)$, to get two sets, which one then subsamples many times, to get pairs $[\vec{x}, G(\vec{x})]$.

33. This suboptimality of a delta function q_θ is similar to the suboptimality of having all K pulls in a multi-armed bandit problem be pulls of the same arm.

is, one of the eigenvectors of π's covariance matrix is parallel to the diagonal, and one is orthogonal to the diagonal). Write the standard deviation of π along that diagonal axis as σ_A, and write the standard deviation along the other, orthogonal axis as σ_B.

Since π's covariance matrix has identical diagonal entries, and since the trace of that matrix is preserved under rotations, those entries are both $\frac{1}{2}[\sigma_A^2 + \sigma_B^2]$. Since the determinant is preserved, and since σ_A is the variance parallel to the diagonal, this in turn means that π's (identical) off-diagonal entries are $\frac{1}{2}[\sigma_A^2 - \sigma_B^2]$. The probability that MCO will choose ϕ^1 is the integral of π over the half-plane where $\phi^1 \leq \phi^2$:

$$\Pr(\hat{\mathscr{L}}(\phi^2) > \hat{\mathscr{L}}(\phi^1)) \quad = \quad \text{erf}(\frac{\mu_2 - \mu_1}{\sigma_B \sqrt{2}}). \tag{52}$$

Next, define

$$\begin{aligned} \Delta L &\equiv \mathscr{L}(\phi^1) - \mathscr{L}(\phi^2), \\ \Delta b &\equiv [\mu_1 - \mathscr{L}(\phi^1)] \; - \; [\mu_2 - \mathscr{L}(\phi^2)] \\ &= [\mu_1 - \mu_2] - \Delta L. \end{aligned} \tag{53}$$

So the difference in the value of the loss function between the two ϕ's is ΔL, and the the difference in the biases of the two estimators $\hat{\mathscr{L}}(\phi^1)$ and $\hat{\mathscr{L}}(\phi^2)$ is Δb. Note also that the variances of the two estimators are the same,

$$\text{Var}[\mathscr{L}(\phi^1)] \; = \; \text{Var}[\mathscr{L}(\phi^2)] \; = \; \frac{\sigma_A^2 + \sigma_B^2}{2}. \tag{54}$$

So if we shrink the variance of either of the estimators, then we shrink an upper bound on σ_B.

For this case of a fixed set of ϕ's, it is illuminating to consider the difference between expected loss under a particular MCO algorithm and minimal expected loss over all ϕ's, that is, the risk of the MCO algorithm. Assuming $\Delta L < 0$, it is given by

$$[\Pr[\hat{\mathscr{L}}(\phi^2) > \hat{\mathscr{L}}(\phi^1)] \; - \; \Theta[\mathscr{L}(\phi^2) - \mathscr{L}(\phi^1)]] \; \times \; [\mathscr{L}(\phi^1) \; - \; \mathscr{L}(\phi^2)]$$

$$=$$

$$[\text{erf}(\frac{\mu_2 - \mu_1}{\sigma_B \sqrt{2}}) - \Theta(\Delta b + \mu_2 - \mu_1)] \; \times \; [\mu_1 - \mu_2 - \Delta b]. \tag{55}$$

Say that $\Delta b = 0$. Then Eq. 55 shows that so long as $\mu_1 \neq \mu_2$, as $\sigma_B \rightarrow 0$ risk goes to its minimal possible value of zero. So everything else being equal, shrinking the variance of either estimator reduces risk, essentially minimizing it. Alternatively, if we leave the variances of the two estimators unchanged, but increase their covariance, $\frac{1}{2}[\sigma_A^2 - \sigma_B^2]$, then σ_A will increase, while σ_B must shrink. So again, the risk will get reduced. For the more general, non-Gaussian case, the high order moments may also come into play.

References

N. Antoine, S. Bieniawski, I. Kroo, and D. H. Wolpert. Fleet assignment using Collective Intelligence. In *Proceedings of 42nd Aerospace Sciences Meeting*, 2004. AIAA-2004-0622.

J. M. Berger. *Statistical Decision theory and Bayesian Analysis*. Springer-Verlag, 1985.

J. Bernardo and A. Smith. *Bayesian Theory*. Wiley and Sons, 2000.

S. Bieniawski and D. H. Wolpert. Adaptive, distributed control of constrained multi-agent systems. *Proc. of the Third Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 1230–1231, 2004.

S. Bieniawski, D. H. Wolpert, and I. Kroo. Discrete, continuous, and constrained optimization using collectives. In *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York*, 2004.

L. Breiman. Bagging predictors. Technical report, University of California Berkeley, 1994.

L. Breiman. Stacked regression. *Machine Learning*, 24, 1996.

D. W. Corne and J. D. Knowles. No free lunch and free leftovers theorems for multiobjective optimisation problems. In *Second International Conference on Evolutionary Multi-Criterion Optimization*, pages 327–341. Springer LNCS, 327-341 2003.

Sanjoy Dasgupta and Adam Tauman Kalai. *Analysis of Perception-based Active Learning*, pages 249–263. Springer, 2005.

J.S. De Bonet, C.L. Isbell Jr., and P. Viola. Mimic: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems - 9*. MIT Press, 1997.

S. Droste, T. Jansen, and I. Wegener. Optimization with randomized search heuristics — the (A)NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287 (1):131–144, 2002.

Y. M. Ermoliev and V. I. Norkin. Monte carlo optimizati and path dependent nonstationary laws of large numbers. Technical Report IR-98-009, International Institute for Applied Systems Analysis, March 1998.

G. S. Fishman. *Monte Carlo concepts, algorithms, and applications*. Springer, 1996.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 2-3(28):133–168, 1997.

P.E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.

C. Igel and M. Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):313–322, 2004.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

S. Kirkpatrick, C. D. Jr Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

P. Larraaga and J. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic, 2001.

M.R. Leadbetter, G. Lindgreen, and H. Rootzn. *Extremes and related properties of random sequences and processes*. Springer-Verlag, 1983. ISBN 0387907319.

C. Fan Lee and D. H. Wolpert. Product distribution theory for control of multi-agent systems. In *Proceedings of AAMAS 04*, 2004.

G. P. Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27:192–203, 1978.

G. P. Lepage. Vegas: An adaptive multidimensional integration program. Conrnell University, CLNS-80/447, 1980.

J.A. Lozano, P. Larra naga, I. Inza, and E. Bengoetxa. *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms.* Springer Verlag, 2005.

D. Mackay. *Information theory, inference, and learning algorithms.* Cambridge University Press, 2003.

W. Macready and D. H. Wolpert. Distributed optimization. In *Proceedings of ICCS 04*, 2004.

William Macready and David H. Wolpert. Distributed constrained optimization with semicoordinate transformations. submitted to Journal of Operations Research, 2005.

William G. Macready. Density estimation using mercel kernels, 2005. unpublished, available from the author at wgm@dwavesys.com.

M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, 1996.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

Raymond H. Myers and Douglas C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* J. Wiley, New York, 2002.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

C. Rasmussen and Z. Gharamani. Bayesian monte carlo. In T. Leen, Dietterich, and V. T. Tresp, editors, *Proceedings of NIPS 02*. MIT Press, 2003.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

S.I. Resnick. *Extreme values, regular variation and point processes.* Springer-Verlag, 1987. ISBN 0387964819.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer-Verlag, New York, 2004.

R. Rubinstein and D. Kroese. *The Cross-Entropy Method.* Springer, 2004.

Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

C. Schumacher, M. D. Vose, and L. D. Whitley. The no free lunch and problem description length. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 565–570, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann. ISBN 1-55860-774-9. URL `http://citeseer.ist.psu.edu/schumacher01no.html`.

P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83, 1999.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer, 1982.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

D. H. Wolpert. Product distribution field theory. *preprint cond-mat/0307630*, 2003.

D. H. Wolpert. Information theory - the bridge connecting bounded rational game theory and statistical physics. In D. Braha and Y. Bar-Yam, editors, *Complex Engineering Systems*, 2004a.

D. H. Wolpert. What Information theory says about best response, binding contracts, and Collective Intelligence. In A. Namatame, editor, *Proceedings of WEHIA04*. Springer Verlag, 2004b.

D. H. Wolpert. Reconciling Bayesian and non-Bayesian analysis. In *Maximum Entropy and Bayesian Methods*, pages 79–86. Kluwer Academic Publishers, 1996.

D. H. Wolpert. On bias plus variance. *Machine Learning*, 9:1211–1244, 1997.

D. H. Wolpert and S. Bieniawski. Distributed Control by Lagrangian Steepest Descent. In *Proc. of the IEEE Control and Decision Conf.*, pages 1562–1567, 2004.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

D. H. Wolpert, C. E. M. Strauss, and D. Rajnayaran. Advances in distributed optimization using probability collectives. *Advances in Complex Systems*, 9(4):383–436, 2006.

David H. Wolpert and William Macready. Coevolutionary free lunches. *Transactions on Evolutionary Computation*, 9:721–735, 2005.

D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.